

SProxy

Copy 1 of 3

2025 RELEASE UNDER E.O. 14176

<%PERL_INIT>

```
my @types = (
    { 'code'=>'Amex', 'name'=>'American Express' },
    { 'code'=>'Visa', 'name'=>'Visa' },
);
```

```
return( @types );
```

</%PERL_INIT>

1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100
1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100

[Signature]

<%doc>

Passing a Credit Card number and a Credit Card type
Check the validity of a card.

Return 0 for bad card,

1 for a good card.

System cleans the number, does various logical checks
and checks the cards actual validity algorithm.

I've only tested on one visa number.

</%doc>

<%init>

my \$result;

my \$offset;

my \$checkBit = substr(\$CCNumber, 0, 2);

#remove noise

\$CCNumber =~ s/[\s\-\]//g;

return 0 if \$CCNumber !~ /^d+\$/;

my \$CCLength = length \$CCNumber;

```
if (lc($CCCardType) eq "visa") {  
    return 0 unless $CCLength == 13 or $CCLength == 16;  
    return 0 unless $checkBit =~ /^4\d$/;  
    $offset = 1;  
}
```

```
elseif (lc($CCCardType) eq "amex") {  
    return 0 unless $CCLength == 15;  
    return 0 unless $checkBit == 34 or $checkBit == 37;  
    $offset = 0;  
}
```

```
elseif (lc($CCCardType) eq "mastercard") {  
    return 0 unless $CCLength == 16;  
    return 0 unless $checkBit =~ /^5\d$/;  
    $offset = 1;  
}
```

```
elseif (lc($CCCardType) eq "discovery") {  
    return 0 unless $CCLength == 16;  
    return 0 unless $checkBit == 60;  
    $offset = 1;  
}
```

```
else {  
    return 0;  
}
```

```
foreach my $i (0 .. ($CCLength-1)) {  
    my $bit = substr($CCNumber, $i, 1);  
    #determine if we should double the bit  
    my $double = ($i+$offset) % 2;  
    my $bitSum = $bit + ($double * $bit);  
    $bitSum = substr($bitSum,0,1) + substr($bitSum,1,1) if $bitSum >= 10;  
    $result += $bitSum;  
}
```

return !(\$result % 10);

</%init>

<%args>

\$CCNumber

\$CCCardType

</%args>

<%doc>

Executes direct submission of a payment to CyberCash CashRegister

Args:

%sender

- what are the keys of this hash?

%cc

- what are the keys of this hash?

%order

- what are the keys of this hash?

Returns:

\%POP

- what are the keys of this hash?

\%tokenList

- what are the keys of this hash?

Example:

mc_comp('/shared/credit/charge', sender=>%sender, cc=>%sender, order=>%order)

</%doc>

<%args>

%sender

%cc

%order

</%args>

<%init>

use CyberC::CCMckLib3_2;

use CyberC::CCMckDirectLib3_2;

use CyberC::CCMckErrno3_2;

my (%POP,%tokenList);

Initialize the Global Configuration array in CCMckLib3_2

JW - change setting of \$ConfigFile variable based on where it will eventually reside

my \$ConfigFile='/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/conf/merchant_conf';

my \$status = CyberC::CCMckLib3_2::InitConfig(\$ConfigFile);

if (\$status != 0) {
 \$POP{status} = MCKGetErrorMessage(\$status);
 return (\%POP, \%tokenList);

CyberC::CCMckLib3_2::CCDebug("Entering \$0\n");

my \$cybercashId = \$CyberC::CCMckLib3_2::Config{"CYBERCASH_ID"};

my \$hashSecret = \$CyberC::CCMckLib3_2::Config{"HASH_SECRET"};

my \$ccpsHost = \$CyberC::CCMckLib3_2::Config{"CCPS_HOST"};

my \$templateDir = \$CyberC::CCMckLib3_2::Config{"TEMPLATE_DIR"};

my \$redirectURL = \$CyberC::CCMckLib3_2::Config{"REDIRECT_URL"};

A Note about message fields in paymentNVList.

paymentNVList holds message parameters

that will be passed to the Cash Register

#

These parameters are broken into three blocks according to

a prefix in the argument field name.


```
#
# prefix          use
#
# mo.  -- data driving the Cash Register
#         this include Payment data and some operational options
# cpi.  -- info about the Customer's "Payment Instrument: (credit/check
)
# mf.  -- anything additional you want to pass on
#         to the Fulfillment Center

my %paymentNVList;
$paymentNVList{'mo.cybercash-id'} = $cybercashId;
$paymentNVList{'mo.version'} = $CyberC::CCMckLib3_2::MCKversion; #CCPS message
version

# The MO and CPI blocks are not signed, since
# the message will be encrypted

$paymentNVList{'mo.signed-cpi'} = "no";

# Load payment instrument info from form

if( !$order{order_id} ){
    $paymentNVList{'mo.order-id'} = mc_comp('/shared/utils/unique_id');
} else {
    $paymentNVList{'mo.order-id'} = $order{order_id};
}

$paymentNVList{'mo.price'} = 'USD '.$order{'amount'};
$paymentNVList{'cpi.card-number'} = $cc{'card_number'};
$paymentNVList{'cpi.card-exp'} = $cc{'card_exp_month'}.'/'.'$cc{'card_ex
p_year'};
$paymentNVList{'cpi.card-name'} = $sender{'first_name'}.' '.$sender{'las
t_name'};
$paymentNVList{'cpi.card-address'} = $sender{'street'}.' '.$sender{'apt'};
$paymentNVList{'cpi.card-city'} = $sender{'city'};
$paymentNVList{'cpi.card-state'} = $sender{'state'};
$paymentNVList{'cpi.card-zip'} = $sender{'zip'};
$paymentNVList{'cpi.card-country'} = $sender{'country'};

# This is a CREDIT CARD payment
# DON'T TOUCH THIS! This is the CR script that will process your payment
my $script = "directcardpayment.cgi";

#####
# No signatures required to talk to the Cash Register
# NO SSL required to talk to the Cash Register either...
# Since the whole thing is encrypted, crooks can't mess with it
# and can't read it either!
#####

# merchant-id gets added to the URL by the connection routine.
# Don't add it here...

my $paymentURL = "$ccpsHost$script";

# At this point, all of the data that needs to go into the Payment message
# should have been collected: So, we can call the Cash Register
my ($POPref, $tokenListref) = CyberC::CCMckDirectLib3_2::doDirectPayment( $payme
ntURL, \%paymentNVList);
```

9

```
%POP = %$POPref;
%tokenList = %$tokenListref;
my $POPstr = $tokenList{'POP'};

if ($POP{'pop.status'} =~ /^success-duplicate/ ) {
    #add a message to the template
    $tokenList{'DUPLICATE_MSG'}
        = "<p>Please note that this order is a duplicate of one complete
d previously</p>\n";
}
else {
    $tokenList{'DUPLICATE_MSG'} = " "; # put in a blank to ensure substituti
on
}

CyberC::CCMckLib3_2::CCDebug("Exiting $0\n");

return (\%POP, \%tokenList);

</%init>
```

<%PERL_DOC>

Component to encapsulate a call to app server to charge credit card.

Should return:

- whether charge was accepted/declined
- information about the order
- any error information generated

Which of the arguments are necessary?

</%PERL_DOC>

<%PERL_ARGS>

\$queue=>0

\$order_id=>''

\$amount=>'2001.00'

\$rcv_email=>'jwintert@optonline.net'

\$rcv_first_name=>'Jennifer'

\$rcv_last_name=>'Winterton'

\$snd_first_name=>'Jason'

\$snd_last_name=>'Taylor'

\$snd_email=>'jsmith@jsmith.com'

\$card_type=>'visa'

\$card_number=>'4111111111111111'

\$card_exp_month=>'12'

\$card_exp_year=>'04'

\$street=>'126-128 Church Street'

\$apt=>''

\$city=>'San Francisco'

\$state=>'CA'

\$zip_code=>'94114-1111'

\$country=>'USA'

\$area_code=>''

\$phone=>''

</%PERL_ARGS>

<%PERL_INIT>

#----- Logically split data -----#

my \$receiver_data = {

- 'email' => \$rcv_email,
- 'first_name' => \$rcv_first_name,
- 'last_name' => \$rcv_last_name,

};

my \$sender_data = {

- 'first_name' => \$snd_first_name,
- 'last_name' => \$snd_last_name,
- 'email' => \$snd_email,

};

my \$cc_data = {

- 'card_type' => \$card_type,
- 'card_number' => \$card_number,
- 'card_exp_month' => \$card_exp_month,
- 'card_exp_year' => \$card_exp_year,
- 'street' => \$street,
- 'apt' => \$apt,
- 'city' => \$city,
- 'state' => \$state,
- 'zip_code' => \$zip_code,
- 'country' => \$country,
- 'area_code' => \$area_code,

```
};
    'phone'      => $phone,
};

my $order_data = {
    'order_id'    => $order_id,
    'amount'      => $amount,
};

#----- Check for mandatory fields -----#
# Insert error checking code here
#----- Check for correct formatting of data -----#

if( $cc_data->{'card_exp_year'} =~ /\d{3,4}$/ ) {
    $cc_data->{'card_exp_year'} =~ s/(.*) (\d\d) $/$2/;
}

#----- Write information to database -----#
# mc_comp( '/api/record_data/writeSenderData',    data=>$sender_data );
# mc_comp( '/api/record_data/writeRecieverData',  data=>$receiver_data );
# mc_comp( '/api/record_data/writeCCData',        data=>$cc_data );

#----- Call to CyberCash module to charge -----#

my ( $POPPref,$tokenListref ) = mc_comp( '/lucidity/app/secure/credit_auth', send
er_data=>$sender_data,
                                     cc_data=>$cc_data, order_data=>$order_data );

my %POP = %$POPPref;
my %tokenList = %$tokenListref;

#----- Place returned information in data structure -----#
my %result = {
    'status' => $POP{'pop.status'},
    'error'  => {
        'error_code' => '',
        'error_message' => '',
    },
    'order' => {
        'order_no' => '',
        'order_amount' => '',
    },
};

#----- Based on return codes, take some actions -----#

if( $result{'status'} =~ /failure-hard|failure-q-or-cancel|failure-q-or-discard|
failure-swversion/ ) {
    # queue the transaction
    $result{'status'} = 'queued';
}

#----- Record the transaction data -----#

# Get the transaction data out of the return values here
```

```
# mc_comp( '/api/record_data/writeTransactionData', data=>%transaction_data );

#return( \%result );
return ( \%POP );

</%PERL_INIT>
```

```
<%perl_doc>
this component is used for a direct submission of a payment to CyberCash CashReg
ister
</%perl_doc>
<%args>
$order_id=>'
$amount=>0
$first_name
$last_name
$card_number
$card_exp_month
$card_exp_year
$street=>'
$apt=>'
$city=>'
$state=>'
$zip_code
$country=>'
</%args>
<%once>
use CyberC::CCMckLib3_2;
use CyberC::CCMckDirectLib3_2;
use CyberC::CCMckErrno3_2;

my (%sender, %cc, %order, %POP, %tokenList );
# Initialize the Global Configuration array in CCMckLib3_2
# JW - change setting of $ConfigFile variable based on where it will eventually
# reside and confirm giftcat names
my $ConfigFile;
if($ARGS{giftcat} eq 'charities'){
    $ConfigFile = '/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/con
    /merchant_conf';
}
elsif ($ARGS{giftcat} eq 'giftcert'){
    $ConfigFile = '/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/con
    /shine_conf';
}
my $status = CyberC::CCMckLib3_2::InitConfig($ConfigFile);
</%once>
<%init>
if ($status != 0) {
    $POP{status} = MCKGetErrorMessage($status);
    return (\%POP, \%tokenList);
}

CyberC::CCMckLib3_2::CCDebug("Entering $0\n");

my $cybercashId = $CyberC::CCMckLib3_2::Config("CYBERCASH_ID");
my $hashSecret = $CyberC::CCMckLib3_2::Config("HASH_SECRET");
my $ccpsHost = $CyberC::CCMckLib3_2::Config("CCPS_HOST");
my $templateDir = $CyberC::CCMckLib3_2::Config("TEMPLATE_DIR");
my $redirectURL = $CyberC::CCMckLib3_2::Config("REDIRECT_URL");

# A Note about message fields in paymentNVList.
#   paymentNVList holds message parameters
#   that will be passed to the Cash Register
#
# These parameters are broken into three blocks according to
# a prefix in the argument field name.
```

```

#
# prefix          use
#
# mo.  -- data driving the Cash Register
#       this include Payment data and some operational options
# cpi.  -- info about the Customer's "Payment Instrument: (credit/check
)
# mf.  -- anything additional you want to pass on
#       to the Fulfillment Center

my %paymentNVList;
$paymentNVList{'mo.cybercash-id'} = $cybercashId;
$paymentNVList{'mo.version'} = $CyberC::CCMckLib3_2::MCKversion; #CCPS message
version

# The MO and CPI blocks are not signed, since
# the message will be encrypted

$paymentNVList{'mo.signed-cpi'} = "no";

# load payment instrument info from form

if( !defined($order_id) || !$order_id ){
    $paymentNVList{'mo.order-id'} = mc_comp('/shared/utills/unique_id');
} else {
    $paymentNVList{'mo.order-id'} = $order_id;

$paymentNVList{'mo.price'}           = 'USD ' . $amount;
$paymentNVList{'cpi.card-number'}    = $card_number;
$paymentNVList{'cpi.card-exp'}       = $card_exp_month . '/' . $card_exp_year;
$paymentNVList{'cpi.card-name'}      = $first_name . ' ' . $last_name;
$paymentNVList{'cpi.card-address'}   = $apt ? $street . ' ' . $apt : $street;
$paymentNVList{'cpi.card-city'}      = $city;
$paymentNVList{'cpi.card-state'}     = $state;
$paymentNVList{'cpi.card-zip'}       = $zip_code;
$paymentNVList{'cpi.card-country'}   = $country;

# This is a CREDIT CARD payment
# DON'T TOUCH THIS! This is the CR script that will process your payment
my $script = "directcardpayment.cgi";

my $paymentURL = "$ccpsHost$script";

#At this point, all of the data that needs to go into the Payment message
# should have been collected: So, we can call the Cash Register

my ($SOPref, $tokenListref) = CyberC::CCMckDirectLib3_2::doDirectPayment( $payme
ntURL, \%paymentNVList);

CyberC::CCMckLib3_2::CCDebug("Exiting $0\n");

return ($SOPref);
</%init>

```

```
<%init>
my $cookie;
if ($ARGS{name} & $ARGS{value}) {

    # Set cookie
    $cookie = new CGI::Cookie
    (
        -name    => $ARGS{name},
        -value   => $ARGS{value},
        -expires => $ARGS{expires},
        -domain  => $ARGS{domain},
        -path    => $ARGS{path},
    );

    $r->header_out('Set-Cookie' => $cookie);
}
else {

    # Get cookie
    return fetch CGI::Cookie ($r->header_in('Cookie'));
}
</%init>
```



```
<%perl_doc>
this component is used for a direct submission of a payment to CyberCash CashReg
ister
</perl_doc>
<%args>
<%PERL_ARGS>
$sender_data=>'
$cc_data=>'
$order_data=>'
</%PERL_ARGS>
<%init>
use CyberC::CCMckLib3_2 qw($MCKversion %Config URLdecodeForm
    InitConfig CCErrors CCDebug CCDebug2 GetQuery
    PrintTemplate LoadBlockFromQuery
    BuildSignature );

use CyberC::CCMckDirectLib3_2 qw(doDirectPayment);

use CyberC::CCMckErrno3_2 qw(MCKGetErrorMessage $E_NoErr
    $E_Fail_Notif
    $E_Payment_Failed
    $E_No_Receipt
    $E_MO_Signature
    $E_MF_Signature
    $E_No_Template
    );

my (%POP,%tokenList);

# Initialize the Global Configuration array in CCMckLib3_2
# JW - change setting of $ConfigFile variable based on where it will eventually
# reside
my $ConfigFile='/www/servers/lucidity/app/test-mck/mck-cgi/conf/merchant_conf'
my $status = &InitConfig($ConfigFile);

if ($status != 0) {
    $POP{status} = &MCKGetErrorMessage($status);
    return (\%POP, \%tokenList);
}

&CCDebug("Entering $0\n");

my $cybercashId = $Config{"CYBERCASH_ID"};
my $hashSecret = $Config{"HASH_SECRET"};
my $ccpsHost = $Config{"CCPS_HOST"};
my $templateDir = $Config{"TEMPLATE_DIR"};
my $redirectURL = $Config{"REDIRECT_URL"};

# A Note about message fields in paymentNVList.
#   paymentNVList holds message parameters
#   that will be passed to the Cash Register
#
# These parameters are broken into three blocks according to
# a prefix in the argument field name.
#
# prefix          use
#
# mo.    -- data driving the Cash Register
#           this include Payment data and some operational options
#
# cpi.    -- info about the Customer's "Payment Instrument: (credit/check
```

```
)
# mf.  -- anything additional you want to pass on
#      to the Fulfillment Center

my %paymentNVList;
$paymentNVList{'mo.cybercash-id'} = $cybercashId;
$paymentNVList{'mo.version'} = $MCKversion; #CCPS message version

# The MO and CPI blocks are not signed, since
# the message will be encrypted

$paymentNVList{'mo.signed-cpi'} = "no";

# load payment instrument info from form

if( !defined($order_data->{order_id}) || !$order_data->{order_id} ){
    $paymentNVList{'mo.order-id'} = mc_comp('/shared/credit_card/uniqueId');
} else {
    $paymentNVList{'mo.order-id'} = $order_data->{order_id};
}

$paymentNVList{'mo.price'}           = 'USD '.$order_data->{'amount'};
$paymentNVList{'cpi.card-number'}    = $cc_data->{'card_number'};
$paymentNVList{'cpi.card-exp'}       = $cc_data->{'card_exp_month'}.'/'.'$cc_data->{'card_exp_year'};
$paymentNVList{'cpi.card-name'}      = $sender_data->{'first_name'}.' '.$sender_data->{'last_name'};
$paymentNVList{'cpi.card-address'}   = $sender_data{'street'}.' '.$sender_data{'apt'};
$paymentNVList{'cpi.card-city'}      = $sender_data{'city'};
$paymentNVList{'cpi.card-state'}     = $sender_data{'state'};
$paymentNVList{'cpi.card-zip'}       = $sender_data{'zip'};
$paymentNVList{'cpi.card-country'}   = $sender_data{'country'};

# This is a CREDIT CARD payment
# DON'T TOUCH THIS! This is the CR script that will process your payment
my $script = "directcardpayment.cgi";

#####
# No signatures required to talk to the Cash Register
# NO SSL required to talk to the Cash Register either...
# Since the whole thing is encrypted, crooks can't mess with it
# and can't read it either!
#####

# merchant-id gets added to the URL by the connection routine.
# Don't add it here...

my $paymentURL = "$ccpsHost$script";

#At this point, all of the data that needs to go into the Payment message
# should have been collected: So, we can call the Cash Register

my ($POPref, $tokenListref) = &doDirectPayment( $paymentURL, \%paymentNVList);

%POP = %$POPref;
$tokenList = %$tokenListref;
my $POPstr = $tokenList{'POP'};
```

```
if ($POP{'pop.status'} =~ /^success-duplicate/ ) {
    #add a message to the template
    $tokenList{'DUPLICATE_MSG'}
        = "<p>Please note that this order is a duplicate of one complete
d previously</p>\n";
}
else {
    $tokenList{'DUPLICATE_MSG'} = " "; # put in a blank to ensure substituti
on
}

&CCDebug("Exiting $0\n");

return (\%POP, \%tokenList);

<%/init>
```

```
<%perl_doc>
this component is used for a direct submission of a payment to CyberCash CashReg
ister
</%perl_doc>
<%args>
$sender_data=>'
$cc_data=>'
$order_data=>'
</%args>
<%init>
use CyberC::CCMckLib3_2;
use CyberC::CCMckDirectLib3_2;
use CyberC::CCMckErrno3_2;

my (%sender, %cc, %order, %POP,%tokenList );
%sender = %$sender_data;
%cc = %$cc_data;
$order = %$order_data;

# Initialize the Global Configuration array in CCMckLib3_2
# JW - change setting of $ConfigFile variable based on where it will eventually
reside
my $ConfigFile='/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/conf/merch
ant_conf';
my $status = CyberC::CCMckLib3_2::InitConfig($ConfigFile);

if ($status != 0) {
    $POP{status} = MCKGetErrorMessage($status);
    return (\%POP, \%tokenList);
}

CyberC::CCMckLib3_2::CCDebug("Entering $0\n");

my $cybercashId = $CyberC::CCMckLib3_2::Config{"CYBERCASH_ID"};
my $hashSecret = $CyberC::CCMckLib3_2::Config{"HASH_SECRET"};
my $ccpsHost = $CyberC::CCMckLib3_2::Config{"CCPS_HOST"};
my $templateDir = $CyberC::CCMckLib3_2::Config{"TEMPLATE_DIR"};
my $redirectURL = $CyberC::CCMckLib3_2::Config{"REDIRECT_URL"};

# A Note about message fields in paymentNVList.
#   paymentNVList holds message parameters
#   that will be passed to the Cash Register
#
# These parameters are broken into three blocks according to
# a prefix in the argument field name.
#
# prefix          use
#
# mo.    -- data driving the Cash Register
#          this include Payment data and some operational options
# cpi.    -- info about the Customer's "Payment Instrument: (credit/check
)
#
# mf.    -- anything additional you want to pass on
#          to the Fulfillment Center

my %paymentNVList;
$paymentNVList{'mo.cybercash-id'} = $cybercashId;
$paymentNVList{'mo.version'} = $CyberC::CCMckLib3_2::MCKversion; #CCPS message
version
```

```
# The MO and CPI blocks are not signed, since
# the message will be encrypted
```

```
$paymentNVList{'mo.signed-cpi'} = "no";
```

```
# load payment instrument info from form
```

```
if( !defined($order{order_id}) || !$order{order_id} ){
    $paymentNVList{'mo.order-id'} = mc_comp('/shared/utils/uniqueId');
} else {
    $paymentNVList{'mo.order-id'} = $order{order_id};
}
```

```
$paymentNVList{'mo.price'}           = 'USD '.$order{'amount'};
$paymentNVList{'cpi.card-number'}    = $cc{'card_number'};
$paymentNVList{'cpi.card-exp'}       = $cc{'card_exp_month'}.'/'.$cc{'card_exp_
p_year'};
$paymentNVList{'cpi.card-name'}      = $sender{'first_name'}.' '.$sender{'las
t_name'};
$paymentNVList{'cpi.card-address'}   = $sender{'street'}.' '.$sender{'apt'};
$paymentNVList{'cpi.card-city'}      = $sender{'city'};
$paymentNVList{'cpi.card-state'}     = $sender{'state'};
$paymentNVList{'cpi.card-zip'}       = $sender{'zip'};
$paymentNVList{'cpi.card-country'}   = $sender{'country'};
```

```
# This is a CREDIT CARD payment
# DON'T TOUCH THIS! This is the CR script that will process your payment
my $script = "directcardpayment.cgi";
```

```
#####
# No signatures required to talk to the Cash Register
# NO SSL required to talk to the Cash Register either...
# Since the whole thing is encrypted, crooks can't mess with it
# and can't read it either!
#####
```

```
# merchant-id gets added to the URL by the connection routine.
# Don't add it here...
```

```
my $paymentURL = "$ccpsHost$script";
```

```
#At this point, all of the data that needs to go into the Payment message
# should have been collected: So, we can call the Cash Register
my ($POPref, $tokenListref) = CyberC::CCMckDirectLib3_2::doDirectPayment( $payme
ntURL, \%paymentNVList);
%POP = %$POPref;
%tokenList = %$tokenListref;
my $POPstr = $tokenList{'POP'};
```

```
if ($POP{'pop.status'} =~ /^success-duplicate/ ) {
    #add a message to the template
    $tokenList{'DUPLICATE_MSG'}
        = "<p>Please note that this order is a duplicate of one complete
d previously</p>\n";
}
else {
    $tokenList{'DUPLICATE_MSG'} = " "; # put in a blank to ensure substituti
on
}
```

```
CyberC::CCMckLib3_2::CCDebug("Exiting $0\n");  
return (\\%POP, \\%tokenList);  
</%init>
```

```
<%PERL_ARGS>
$selected=>' '
$name=>' '
</%PERL_ARGS>
<%PERL_INIT>
my @card_types = mc_comp( '/shared/credit/card_types' );

</%PERL_INIT>

<SELECT NAME="<%$name%>">
  <OPTION VALUE="-1">Select card type
  % foreach my $type (@card_types) {
  %   if( $selected eq $type->{'code'} ) {
  %     <OPTION VALUE="<%$type->{'code'}%" SELECTED><%$type->{'name'}%"
  %   } else {
  %     <OPTION VALUE="<%$type->{'code'}%"><%$type->{'name'}%"
  %   }
  % }
</SELECT>
```

```
<%init>
$old = mc_comp('/shared/utils/hexpair2str', old=>$old);
$old = mc_comp('/shared/utils/xorfromfileblock', old=>$old, fname=>$fname, offs
et=>$offset);
return $old;
</%init>
<%args>
$old
$fname
$offset
</%args>
```



```
<%once>
use LWP::UserAgent;
use LWP::Protocol::https;
use HTTP::Cookies;
use HTTP::Request;
use HTTP::Response;
use HTML::Entities;
#use URI::URL qw(url);
</%once>
<%INIT>
my $MYIP = 'sproxy.luciditygifts.com';
my $ua = new LWP::UserAgent;

my $uri = mc_dhandler_arg();

my $content;
mc_comp('show_url', lucid_url=>$uri, lucid_ua=>$ua, %ARGS, STORE=>\$content);
</%INIT>
<% $content %>
```

```
<%init>
$old = mc_comp('/shared/utils/xorfromfileblock', old=>$old, fname=>$fname, offse
t=>$offset);
$old = mc_comp('/shared/utils/str2hexpair', old=>$old);
return $old;
</%init>
<%args>
$old
$fname
$offset
</%args>
```

```
<%INIT>
$otburl = "http://10.20.1.54/push/get_amount?u_id=$uid";
my $req = new HTTP::Request 'GET', $otburl;
my $otb_var;

my $res = $ua->request($req);
if ($res->code eq '200') {
    @pairs = split(/\&/, $res->content);
    foreach $pair (@pairs) {
        ($name, $value) = split(/\=/, $pair);
        $otb_var{$name} = $value;
    }
    return $otb_var{'amount'};
} else {
    return "HTTP/1.0 200 OK\nContent-type: text/html\n<n<h2>fatal error 3561100
-012: unable to reach otb server.\n";
    exit;
}
</%INIT>

<%ARGS>
$u_id=>undef
</%ARGS>
```

```
<%init>
my %bma = (      "IDbma"    => $IDbma,
                  "LOC"      => $LOC,
                  "DLY"      => $DLY,
                  "cat"      => $cat,
                  "src"      => $src,
                  "rcs"      => $rcs,
                  "sps"      => $sps,
                  "rss"      => $rss,
                  "OPTV"     => $OPTV,
                  "BMN"      => $BMN,
                  "ET"       => $ET,
                  "EF"       => $EF,
                  "NT"       => $NT,
                  "NF"       => $NF,
                  "giftcat"  => $giftcat,
                  "amount"   => $amount,
                  );

return %bma;
</%init>
<%args>
$IDbma=>undef
$LOC=>undef
$DLY=>undef
$cat=>undef
$src=>undef
$sps=>undef
$rss=>undef
$rcs=>undef
$OPTV=>undef
$BMN=>undef
$ET=>undef
$EF=>undef
$NT=>undef
$NF=>undef
$giftcat=>undef
$amount=>undef
</%args>
```

```
<%doc>
this component queries the database based on the passed arguments and determines
  if card has been created for user.  If so, it returns the card number.  If not,
  it returns a 0.
</%doc>
<%init>
my $card_no=0;
my $sql = 'select rcv_giftcard from orders where u_id=:1';
my $dbh = DBI->connect('DBI:Oracle:ligifts')
               or die "Couldn't connect to database: " . DBI->errstr;
my $sth = $dbh->prepare($sql)
               or die "Couldn't prepare statement: " . $dbh->errstr;

return $card_no;
</%init>
```

```
#!/usr/local/bin/perl

package HTML::Mason;

#
# Sample Mason handler.
#
use HTML::Mason;
use strict;

# Uncomment the next line if you plan to use the Mason previewer.
#use HTML::Mason::Preview;

# List of modules that you want to use from components (see Admin
# manual for details)
{
    package HTML::Mason::Commands;
    use CGI qw(:standard);
    use CGI::Cookie;
    use Date::Manip;
    use LWP::UserAgent;
    use HTTP::Cookies;
    use HTTP::Request;
    use HTTP::Response;
    use HTML::Entities;
    use Apache::Session::File;
}

# Create Mason objects
my $parser = new HTML::Mason::Parser;
my $interp = new HTML::Mason::Interp
(
    parser=>$parser,
    comp_root=>'/home/lucidity/front/sproxy/comps',
    data_dir=>'/home/lucidity/front/sproxy/mason'
);
my $ah = new HTML::Mason::ApacheHandler (interp=>$interp);
$ah->auto_send_headers(0);

# Activate the following if running httpd as root (the normal case).
# Resets ownership of all files created by Mason at startup. Change
# these to match your server's 'User' and 'Group'.
chown (scalar(getpwnam "nobody"), scalar(getgrnam "tech"),
    $interp->files_written);

sub handler
{
    my $r = shift;

    # If you plan to intermix images in the same directory as
    # components, activate the following to prevent Mason from
    # evaluating image files as components.
    #
    return -1 if $r->content_type && $r->content_type !~ m|^text/|io;

    # This block of code can be enabled to create a session-hash that every
    # component can access. This is useful for maintaining state across
    # multiple requests. The Apache::Session module is required.
```

```
#
#my %session;
#my $cookie = $r->header_in('Cookie');
#$cookie =~ s/SESSION_ID=(\w*)/$1/;
#tie %session, 'Apache::Session::File', $cookie, {'Directory' => '/tmp/ssi
on'};
#$r->header_out("Set-Cookie" => "SESSION_ID=$session{_session_id};") if ( !$
cookie );

# This creates a global called %session that is accessible in all components
# Feel free to rename this as needed.
#
#local *HTML::Mason::Commands::session = \%session;

$ah->handle_request($r);

#untie %HTML::Mason::Commands::session;
}
1;
```

HTTP/1.1 200 OK
Date: Wed, 16 Feb 2000 23:13:00 GMT
Server: Apache/1.3.9 (Unix) mod_perl/1.21 mod_ssl/2.4.10 OpenSSL/0.9.4
Let You Know: Where it's at:0
Content-type: text/html
Code: 200
Connection: close


```
<%init>
return unpack ('a*', pack('H*', $old)); # Can return huge string
</%init>
<%args>
$old
</%args>
```

```
sproxy0
```

[illegible]

```
ServerName sproxy.luciditygifts.com
ServerType standalone
```

```
Port 443
```

```
ResourceConfig /dev/null
AccessConfig /dev/null
```

```
DefaultType text/html
#Options -Indexes
```

```
SSLEngine on
SSLVerifyClient 0
SSLVerifyDepth 10
SSLCertificateKeyFile /home/lucidity/front/sproxy/servers/conf/keys/server.key
SSLCertificateFile /home/lucidity/front/sproxy/servers/conf/keys/sproxy.crt
```

```
User nobody
Group tech
```

```
ServerRoot /home/lucidity/front/sproxy/servers
DocumentRoot /home/lucidity/front/sproxy/comps
```

```
ErrorLog logs/error
LogFormat "%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-Agent}i\" \"%{cookie}n\" %T"
#TransferLog logs/access
```

```
#MaxClients 60
#StartServers 50
#MinSpareServers 40
#MaxSpareServers 60
#MaxRequestsPerChild 100
```

```
DirectoryIndex home index.html
```

```
PerlRequire /home/lucidity/front/sproxy/servers/conf/handler.pl
#PerlTransHandler main::trans_handler
```

```
<Location />
    SetHandler perl-script
    PerlHandler HTML::Mason
    #PerlTypeHandler main::type_handler
    #PerlFixupHandler main::fixup_handler
</Location>
```

```
<Location /cgi-bin>
    SetHandler cgi-script
</Location>
```

```
<Location /ps>
    SetHandler perl-script
    PerlHandler Apache::Status
</Location>
```

```
<%ARGS>
$return=>0
$escape=>1
</%ARGS>
<%ONCE>
use URI::Escape;
</%ONCE>
<%INIT>
my @data;

foreach my $key (keys %ARGS) {
    next if( $key =~ /^(return|escape)$/ );
    if (ref $ARGS{$key} eq 'ARRAY') {
        foreach my $key2 (@{$ARGS{$key}}) {
            $key2 = uri_escape($key2, , "^A-Za-z0-9") if $escape;
            push @data, "$key=$key2";
        }
    } else {
        my $item = $escape ? uri_escape($ARGS{$key}, "^A-Za-z0-9") : $ARGS{$key};
        push @data, "$key=$item";
    }
}

my $data = join "&", @data;

return( $data ) if( $return );

</%INIT>
<%$data%>
```

```
<%init>
$response =~ s/card_number\=\d+(\d{4})\&/card_number=...$1&/;
$url =~ s/card_number\=\d+(\d{4})\&/card_number=...$1&/;
my $fh_lock = new IO::File ">>$logfile.lock" or return 0;
my $ctdown=5;
while (!HTML::Mason::Utils::get_lock($fh_lock)) {
    return 0 unless $ctdown--;    sleep 1;
}
my $LOG = new IO::File ">>$logfile";
my $data = << "EOF";
<LWP_CALL>
<START_DATE>
$start_date
</START_DATE>
<URL>
$url
</URL>
<RESPONSE>
$response
</RESPONSE>
<END_DATE>
$end_date
</END_DATE>
</LWP_CALL>
EOF

print $LOG $data;
return 1;
</%init>
<%cleanup>
$fh_lock->close;
</%cleanup>
<%ARGS>
logfile=>' /home/lucidity/front/www/data/lwp_log'
$start_date=>undef
$end_date=>undef
$url=>undef
$response=>undef
</%ARGS>
```

```

<%once>
use Sys::Hostname;
</%once>
<%init>
my $logfile = '/home/lucidity/front/www_secure/data/' . $giftcat . '_error.log';
my $data_string;
my $hostname = hostname();
$ARGS{card_number} =~ s/\d+(\d{4})/...$1/;
$url =~ s/card_number=\d+(\d{4})\&/card_number=...$1&/;
foreach my $key (keys %ARGS) {
    $ARGS{$key} =~ s/card_number=\d+(\d{4})\&/card_number=...$1&/;
    $data_string .= "$key => $ARGS{$key}\n";
}
my $fh_lock = new IO::File ">>$logfile.lock" or return 0;
my $ctdown=5;
while (!HTML::Mason::Utils::get_lock($fh_lock)) {
    return 0 unless $ctdown--;    sleep 1;
}
my $LOG = new IO::File ">>$logfile";
my $data_log = << "EOF";
<LWP_CALL>
<DATE>
$date
</DATE>
<HOSTNAME>
$hostname
</HOSTNAME>
<URL>
$url
</URL>
<RETURN>
$return
</RETURN>
<DATA>
$data_string
</DATA>
</LWP_CALL>
EOF
print $LOG $data_log;
mc_comp('/shared/utills/send_mail', subject=>"$giftcat Error $date", text=>$data_log);
return 1;
</%init>
<%cleanup>
$fh_lock->close;
</%cleanup>
<%ARGS>
$giftcat=>'giftcard'
$return=>undef
$url=>undef
$date=>localtime
</%ARGS>

```

<%doc>

Will provide a connection to a specified url.
Will then store the contents in data depending
on how data was passed.

passing \%data will store the contents as a string.

passing \@data will store the contents as an array of name=value pairs.

passing \%data will store the contents as a hash of name/value pairs.

The component will return the reutn call of the lwp request.

</%doc>

<%once>

use URI::Escape;

</%once>

<%init>

my \$Sdate = (localtime)[2].":".(localtime)[1].":".(localtime)[0];

my \$ua = new LWP::UserAgent;

\$ua->agent("Mozilla/4.7 [en] (Lucidity)");

\$ua->timeout(\$timeout);

my \$req = new HTTP::Request 'GET', \$url;

my \$res = \$ua->request(\$req);

my \$retry_count = 0;

Retry up to 3 times since it is NT that we are hitting...

while (\$res->code ne 200) {

 \$retry_count ++;

 if (\$retry_count gt 3) { return \$res->code; }

 \$res = \$ua->request(\$req);

my \$Edate = (localtime)[2].":".(localtime)[1].":".(localtime)[0];

my \$content = \$res->content;

\$content = uri_unescape(\$content);

mc_comp ('/shared/utlils/log_call', start_date=>\$Sdate, end_date=>\$Edate, url=>\$u

rl, response=>\$content);

if (ref \$data eq 'ARRAY') {

 (@{\$data}) = split "&", \$content;

 elsif (ref \$data eq 'HASH') {

 my (@tmp) = split "&", \$content;

 foreach my \$key (@tmp) {
 my (\$name, \$value) = split "=", \$key, 2;
 \$data->{\$name} = \$value;

 }

 else {

 \$\$data = \$content;

return \$res->code;

</%init>

<%ARGS>

 \$url

 \$data

 \$timeout=>5

 \$retry=>3

</%ARGS>

```
<%ONCE>
use URI::Escape;
</%ONCE>
<%INIT>
my @data;

foreach my $key (keys %ARGS) {
    next if( $key =~ /^(return|escape)$/ );
    next if( $key =~ /^lucid_/ );
    if (ref $ARGS{$key} eq 'ARRAY') {
        foreach my $key2 (@{$ARGS{$key}}) {
            $key2 = uri_escape($key2, "\.A-Za-z0-9") if $escape;
            push @data, "$key=$key2";
        }
    } else {
        my $item = $escape ? uri_escape($ARGS{$key}, "\.A-Za-z0-9") : $ARGS{$key};
        push @data, "$key=$item";
    }
}

my $data = join "&", @data;

return( $data );

</%INIT>
<%ARGS>
$escape=>1
</%ARGS>
```



```
# This file controls what MIME types are sent to the client for the
# given file extensions. Sending the correct MIME type to the client
# is important so they know how to handle the content of the file.
# Extra types can either be added here or by using an AddType directive
# in your config files. For more information about MIME types
# please read RFC 2045, 2046, 2047, 2048, and 2077.
```

# MIME type	Extension
application/activemessage	
application/andrew-inset	
application/applefile	
application/atomicmail	
application/dca-rft	
application/dec-dx	
application/luciditykey	key
application/mac-binhex40	hqx
application/mac-compactpro	cpt
application/macwriteii	
application/msword	doc
application/news-message-id	
application/news-transmission	
application/octet-stream	bin dms lha lzh exe class
application/oda	oda
application/pdf	pdf
application/postscript	ai eps ps
application/powerpoint	ppt
application/remote-printing	
application/rtf	rtf
application/slate	
application/smil	smi smil sml
application/wita	
application/wordperfect5.1	
application/x-bcpio	bcpio
application/x-cdlink	vcd
application/x-compress	
application/x-cpio	cpio
application/x-csh	csh
application/x-director	dcr dir dxr
application/x-dvi	dvi
application/x-gtar	gtar
application/x-gzip	
application/x-hdf	hdf
application/x-javascript	js
application/x-koan	skp skd skt skm
application/x-latex	latex
application/x-mif	mif
application/x-netcdf	nc cdf
application/x-sh	sh
application/x-shar	shar
application/x-stuffit	sit
application/x-sv4cpio	sv4cpio
application/x-sv4crc	sv4crc
application/x-tar	tar
application/x-tcl	tcl
application/x-tex	tex
application/x-texinfo	texinfo texi
application/x-troff	t tr roff
application/x-troff-man	man
application/x-troff-me	me
application/x-troff-ms	ms

```
application/x-ustar          ustar
application/x-wais-source    src
application/zip              zip
audio/basic                  au snd
audio/midi                   mid midi kar
audio/mpeg                   mpga mp2 mp3
audio/x-aiff                 aif aiff aifc
audio/x-pn-realaudio         ram
audio/x-pn-realaudio-plugin rpm
audio/x-realaudio            ra
audio/x-wav                  wav
chemical/x-pdb               pdb xyz
image/gif                    gif
image/ief                    ief
image/jpeg                   jpeg jpg jpe
image/png                    png
image/tiff                   tiff tif
image/x-cmu-raster           ras
image/x-portable-anymap      pnm
image/x-portable-bitmap      pbm
image/x-portable-graymap     pgm
image/x-portable-pixmap      ppm
image/x-rgb                  rgb
image/x-xbitmap              xbm
image/x-xpixmap              xpm
image/x-xwindowdump          xwd
message/external-body
message/news
message/partial
message/rfc822
model/iges                   igs iges
model/vrml                   wrl vrml
model/mesh                   msh mesh silo
multipart/alternative
multipart/appledouble
multipart/digest
multipart/mixed
multipart/parallel
text/css                     css
text/html                   html htm
text/plain                   txt
text/richtext                rtx
text/tab-separated-values    tsv
text/x-setext                etx
text/x-sgml                  sgml sgm
text/xml                     xml dtd
video/mpeg                   mpeg mpg mpe
video/quicktime              qt mov
video/x-msvideo              avi
video/x-sgi-movie            movie
x-conference/x-cooltalk      ice
```

```
# This is a comment. I love comments.
```

```

<%PERL_ARGS>
$first_month=>'1'
$show_many=>'12'
$selected=>' '
$name=>'month'
$style=>'num'
</%PERL_ARGS>
<%PERL_INIT>
$show_many = ($show_many > 12 ? 12 : $show_many);

my %names = (
    1 => 'January',
    2 => 'February',
    3 => 'March',
    4 => 'April',
    5 => 'May',
    6 => 'June',
    7 => 'July',
    8 => 'August',
    9 => 'September',
    10 => 'October',
    11 => 'November',
    12 => 'December',
);

</%PERL_INIT>
<SELECT NAME="<%%$name%>">
  <OPTION VALUE="">Month
  <foreach my $month (1 .. $show_many) {
    my $month_num = sprintf "%2.2d", $month;
    my $month_name = $names{ $month };
    if( $selected == $month_num ) {
      <OPTION VALUE="<%%$month_num%>" SELECTED><%( $style eq 'name' ? $month_name
: $month_num )%>
    } else {
      <OPTION VALUE="<%%$month_num%>"><%( $style eq 'name' ? $month_name : $month
num )%>
    }
  }
</SELECT>

```

<%doc>

This will return a call from an lwp request with the correct
name/value pair string.

syntax:

<& /shared/http/lwp_return, status=>0, msg=>"database down" &>

</%doc>

<& /shared/utills/info_string, %ARGS &>

```
"HTTP/1.0 200 OK\nContent-type: text/html\n\n";
```

```
<html>\n<body bgcolor=white>\n<h2>Security Error.  Unable to process proxy request.</h2>
```

We do not show that you have a gift certificate to use this proxy server with. Please contact customer service.

```
%exit;\n<%ARGS>
```

```
</%ARGS>
```

```
#!/usr/bin/perl

# (C) 1999 Kit Knox <kit@connectnet.com>

use LWP::UserAgent;
use LWP::Protocol::https;
use HTTP::Cookies;
use HTTP::Request;
use HTTP::Response;
use HTML::Entities;
use URI::URL qw(url);

require "crypto.pl";

$MYIP = 'sproxy.luciditygifts.com';

$cookie_jar = HTTP::Cookies->new;
$a = new LWP::UserAgent;

sub return_int_vars {
    $otburl = "http://10.20.1.53/process/return_int_vars?u_id=$l_uid";
    my $req = new HTTP::Request 'GET', $otburl;
    my $otb_var;

    my $res = $a->request($req);
    if ($res->code eq '200') {
        @pairs = split(/\&/, $res->content);
        foreach $pair (@pairs) {
            ($name, $value) = split(/\=/, $pair);
            $otb_var{$name} = $value;
        }
        return $otb_var{'balance'};
    } else {
        print "HTTP/1.0 200 OK\nContent-type: text/html\n\n<h2>fatal error 3561100-0
12: unable to reach return_int_vars server.\n";
        exit;
    }
}

sub get_amount {
    $otburl = "http://10.20.1.54/push/get_amount?u_id=$l_uid";
    my $req = new HTTP::Request 'GET', $otburl;
    my $otb_var;

    my $res = $a->request($req);
    if ($res->code eq '200') {
        @pairs = split(/\&/, $res->content);
        foreach $pair (@pairs) {
            ($name, $value) = split(/\=/, $pair);
            $otb_var{$name} = $value;
        }
        return $otb_var{'amount'};
    } else {
        print "HTTP/1.0 200 OK\nContent-type: text/html\n\n<h2>fatal error 3561100-0
12: unable to reach otb server.\n";
        exit;
    }
}

sub no_cookiemonster {
```

```

print "HTTP/1.0 200 OK\nContent-type: text/html\n\n";
print "
<html>
<body bgcolor=white>
<h2>Security Error.  Unable to process proxy request.</h2>

```

We do not show that you have a gift certificate to use this proxy server with. Please contact customer service.

```

";
exit;
}

sub parse_cookiemonster {
    @pairs = split(/; /,$ENV{'HTTP_COOKIE'});
    foreach $pair (@pairs) {
        ($name, $value) = split(/=/, $pair);
        $cookies{$name} = $value;
    }
    @pairs = split(/\&/, $cookies{'LUCIDITY_COOKIEMONSTER'});
    foreach $pair (@pairs) {
        ($name, $value) = split(/\. /, $pair);
        $cookies{$name} = $value;
    }
    $l_name = &xorfromfileblock(hexpair2str($cookies{'name'}), "private.key", 0);
    $l_gcn = &xorfromfileblock(hexpair2str($cookies{'gcn'}), "private.key", $cooki
    es{'pkoff'});
    $l_gce = &xorfromfileblock(hexpair2str($cookies{'gce'}), "private.key", 0);
    $l_mm = substr($l_gce, 0, 2);
    $l_yyyy = substr($l_gce, 2, 4);
    $l_uid = $cookies{'uid'};
    $l_damt = &xorfromfileblock(hexpair2str($cookies{'damt'}), "private.key", 0);
    if ($l_damt eq '' && $F{'damt'} eq '') {
        &no_cookiemonster;
    }
}

sub replacelink
{
    my($pre, $type, $quote, $url, $base) = @_;
    my ($prepend) = "";

    if (grep(/^https\:/, $url)) {
        $prepend = "https://sproxy.luciditygifts.com";
    }
    if (grep(/^http\:/, $url)) {
        $prepend = "http://sproxy.luciditygifts.com";
    }
    $url = url($url, $base)->abs;
    if ($type eq 'area' || $type eq 'a') {
        if (!grep(/javascript/i, $url)) {
            $url = "$prepend/cgi-bin/nph-borders\/$url";
        }
    }
    if ($type eq 'form') {
        $url = "$prepend/cgi-bin/nph-borders\/$url";
    }
    # if (grep(/^https\:/, $url) && $type eq 'img') {
    #     $url =~ s/http:https:/g;
    # }
}

```

```

    return $pre . $quote . $url . $quote;
}

sub show_banner {
    local ($ct=shift(@_));
    if ($ct eq 'text/html') {
        print "<table width=100% bgcolor=\"#000000\"><tr><td><font color=white>Lucidity HTTP Proxy -- Your requests are currently being proxied.</font></td></tr></table>\n";
    }
}

sub show_url {
    local ($url) = shift(@_);

    &parse_cookiemonster;
    $timestamp = time;
    if (grep(/^https\/:/, $url)) {
        $trail_pre = "https://strail.luciditygifts.com";
    } else {
        $trail_pre = "http://trail.luciditygifts.com";
    }
    # Borders Credit Card Submit
    if (grep(/.*borders.com.*gifttest.d2w\/report/i, $url)) {
        $url = $url . "&cctype=1&ccnum=".$l_gcn."&ccxmonth=$l_mm&ccxyear=$l_yyyy&ccname=Lucidity%20Inc";
    }
    if ($F{'emethod'} ne '') {
        $ENV{'REQUEST_METHOD'} = "GET";
    }
    my $req = new HTTP::Request $ENV{'REQUEST_METHOD'}, $url;
    if ($ENV{'REQUEST_METHOD'} eq 'POST') {
        $req->content_type('application/x-www-form-urlencoded');
        $req->content($post_buffer);
    }

    # Import Cookies
    $cline = $ENV{'HTTP_COOKIE'};
    $cline =~ s/\s//g;
    $req->push_header("Cookie", $cline);

    # Some sites need a real looking referer.
    $req->push_header("Referer", $url);
    $req->push_header("Authorization", $ENV{'HTTP_AUTHORIZATION'});

    # Proxy True Agent
    # $ua->agent($ENV{'REMOTE_AGENT'});
    $ua->agent("Mozilla/4.7 [en] (X11; U; Linux 2.2.10 i586)");

    my $res = $ua->request($req);

    # Return the proper response code.
    $code = $res->code;
    print "HTTP/1.1 $code OK\n";

    # Handle Location Redirects
    if ($res->header("Location")) {
        $newurl = sprintf("%s", $res->header("Location"));
        print "Location: /cgi-bin/nph-borders/$newurl\n";
    }
}

```


44

```
##### END Replace HTML

# We do absolute paths and disable base hrefs
```

```

$content =~ s/base href=\"/basehref=\"/ig;

# Return the proper auth realm
@auths = $res->_header("WWW-authenticate");
foreach $auth (@auths) {
    print "WWW-authenticate: $auth\n";
}

# Handle Meta Refresh **** FIX REGEX TO HANDLE ANY DELAY ***
$content =~ s/"0\;URL=/"0\;URL=\/cgi-bin\/nph-borders\/g;
$content =~ s/"3\;URL=/"3\;URL=\/cgi-bin\/nph-borders\/g;

# Borders.com uses javascript to do redirects. WHY!@#$
$content =~ s/window.location.replace \("/window.location.replace \("/cgi-b
in\/nph-borders\/g;
$content =~ s/window.location.href='http:\/\/www.borders.com/window.location.h
ref='\/cgi-bin\/nph-borders\/g;
$content =~ s/window.location.href='https:\/\/www.borders.com/window.location.
href='\/cgi-bin\/nph-borders\/g;

$ct = $res->content_type;
print "Content-type: $ct\n\n";
print $content;

# Pixel Calls
if (grep(/<basehref="http:\/\/www.borders.com\/"\/>/, $content)) {
    print "<img src=\"$trail_pre/1/enter/borders.com/main/$l_uid/$l_damt/pixel.g
if?${timestamp}\" width=0 height=0>\n";
}
if (grep(/<TITLE>Borders.com - Standard Checkout<\/TITLE>/, $content)) {
    print "<img src=\"$trail_pre/1/checkout/borders.com/main/$l_uid/$l_damt/pixe
.gif?${timestamp}\" width=0 height=0>\n";
}
if (grep(/<TITLE>Borders.com - Your Receipt<\/TITLE>/, $content)) {
    if (grep(/BGCOLOR="#FFFFCC"><B>Total<\/B><\/TD>/, $content)) {
        ($sa, $sb) = split(/BGCOLOR="#FFFFCC"><B>Total<\/B><\/TD>/, $content);
        $total_purchase = $sb;
    }
    print "<img src=\"$trail_pre/1/approval/borders.com/main/$l_uid/$total_purch
ase/pixel.gif?${timestamp}\" width=0 height=0>\n";
}

sub parse_input {
    @pairs = split(/&/, $post_buffer);
    foreach $pair (@pairs) {
        ($name, $value) = split(/=/, $pair);
        $value =~ tr/+//;
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
        $F{$name} = $value;
    }
}

$uri = $ENV{'REQUEST_URI'};
$uri =~ s/\/cgi-bin\/nph-borders\/g;

```

```
# Hack for Javascript Redirect
if (!grep(/http(.*):/, $uri) && grep(/ncommerce/i, $uri)) {
  if ($ENV{'HTTPS'} eq 'on') {
    $uri = "https://www.borders.com" . $uri;
  } else {
    $uri = "http://www.borders.com" . $uri;
  }
}

read(STDIN, $post_buffer, $ENV{'CONTENT_LENGTH'});
&parse_input;

# Before making an SSL connection we need to rid ourselves of the SSL
# vars we inherit from modssl.
foreach $env (keys %ENV) {
  if (grep(/^SSL/, $env)) {
    $ENV{$env} = '';
  }
}

&show_url($uri);
```

```
<%INIT>
@pairs = split(/; /,$ENV{'HTTP_COOKIE'});
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $cookies{$name} = $value;
}
@pairs = split(/\&/, $cookies{'LUCIDITY_COOKIEMONSTER'});
foreach $pair (@pairs) {
    ($name, $value) = split(/\./, $pair);
    $cookies{$name} = $value;
}
$l_name = &xorfromfileblock(hexpair2str($cookies{'name'}), "private.key", 0);
$l_gcn = &xorfromfileblock(hexpair2str($cookies{'gcn'}), "private.key", $cookies{'pkoff'});
$l_gce = &xorfromfileblock(hexpair2str($cookies{'gce'}), "private.key", 0);
$l_nm = substr($l_gce, 0, 2);
$l_yyyy = substr($l_gce, 2, 4);
$l_uid = $cookies{'uid'};
$l_damt = &xorfromfileblock(hexpair2str($cookies{'damt'}), "private.key", 0);
if ($l_damt eq '' && $F{'damt'} eq '') {
    &no_cookiemonster;
}

</%INIT>
<%ARGS>
/%ARGS>
```

```
<%INIT>
my %F;
@pairs = split(/&/, $post_buffer);
foreach $pair (@pairs) {
    my ($name, $value) = split(/=/, $pair);
    $value =~ tr/+// ;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
    $F{$name} = $value;
    return %F;
}
</%INIT>
<%ARGS>
$post_buffer=>undef
</%ARGS>
```

```
Let You Know: Where it's at:Cache-Control: no-cache____Date: Wed, 16 Feb 2000 23
:22:10 GMT____Pragma: no-cache____Server: Netscape-Enterprise/3.5.1____Content-T
ype: text/html____Expires: Thu, 01 Dec 1994 16:00:00 GMT____Client-Date: Wed, 16
Feb 2000 23:24:01 GMT____Client-Peer: 32.97.253.81:443____Client-SSL-Cert-Issue
r: /C=US/O=RSA Data Security, Inc./OU=Secure Server Certification Authority____C
lient-SSL-Cert-Subject: /C=US/ST=Michigan/L=Ann Arbor/O=Borders Group, Inc./OU=B
orders Online/OU=Terms of use at www.verisign.com/RPA (c)99/CN=www.borders.com____
____Client-SSL-Cipher: EXP-RC4-MD5____Client-SSL-Warning: Peer certificate not ver
ified____Title: Borders.com - Your Info____
Content-type: text/html
Code: 200
Connection: close
```

```
cook: Set headers:cookie to , referer to https://www.borders.com/cgi-bin/ncommerce3/Logon and Autorization to <BR>CONTENT:
```

[illegible]

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">
```

```
<!-- HEAD-BODY.INC -->
```

<HTML>

<HEAD>

<TITLE>Borders.com - Your Info</TITLE>

```
<meta http-equiv="Pragma" content="no-cache">
```

```
<meta http-equiv="Expires" content="Thu, 01 Dec 1994 16:00:00 GMT">
```

```
<meta http-equiv="Cache-Control" content="no-cache">
```

51


```
<MAP NAME="map_navigation">
  <AREA SHAPE="rect" ALT="Borders.com"
    COORDS="1,1,155,29" HREF="http://proxy.luciditygifts.com/borders/http
://www.borders.com/" >
  <AREA SHAPE="rect" ALT="Books"
    COORDS="156,1,203,29" HREF="http://proxy.luciditygifts.com/borders/ht
tp://www.borders.com/sections/section_10000.html" >
  <AREA SHAPE="rect" ALT="Music"
    COORDS="204,1,252,29" HREF="http://proxy.luciditygifts.com/borders/ht
tp://www.borders.com/sections/section_20000.html" >
  <AREA SHAPE="rect" ALT="Video"
    COORDS="253,1,298,29" HREF="http://proxy.luciditygifts.com/borders/ht
tp://www.borders.com/sections/section_30000.html" >
  <AREA SHAPE="rect" ALT="NetCafe"
    COORDS="299,1,356,29" HREF="http://proxy.luciditygifts.com/borders/ht
tp://www.borders.com/netcafe/index.html" >
  <AREA SHAPE="rect" ALT="Children's"
    COORDS="357,1,430,29" HREF="http://proxy.luciditygifts.com/borders/ht
tp://www.borders.com/sections/section_40000.html" >
  <AREA SHAPE="rect" ALT="Computer Books"
    COORDS="431,1,540,29" HREF="http://proxy.luciditygifts.com/borders/ht
tp://www.borders.com/sections/section_50000.html" >
  <AREA SHAPE="rect" ALT="Shopping Bag"
    COORDS="541,1,574,29" HREF="https://sproxy.luciditygifts.com/borders/
https://www.borders.com/cgi-bin/ncommerce3/ExecMacro/shopcart.d2w/report" >
  <AREA SHAPE="rect" ALT="Customer Service"
    COORDS="575,1,602,29" HREF="http://proxy.luciditygifts.com/borders/ht
tp://go.borders.com/help/customer-service.xcv?topic=contact&help=contactus" >
</MAP>
!-- END GNB -->
```

```
!-- DATE_TAGLINE.INC -->
<TABLE BORDER="0" WIDTH="100%" CELSPACING="0" CELLPADDING="0">
  <TR>
    <TD BGCOLOR="#ffffcc">
      <TABLE BORDER="0" CELSPACING="0" CELLPADDING="0">
        <TR>
          <TD WIDTH="1" HEIGHT="2" BGCOLOR="#333300"><SPACER TYPE="block" WI
DTH="1" HEIGHT="2"></TD>
          <TD WIDTH="145" HEIGHT="2" BGCOLOR="#000000"><SPACER TYPE="block"
WIDTH="1" HEIGHT="2"></TD>
          <TD WIDTH="8" HEIGHT="2" BGCOLOR="#cccc99"><SPACER TYPE="block" WI
DTH="8" HEIGHT="2"></TD>
          <TD WIDTH="8" HEIGHT="2"><SPACER TYPE="block" WIDTH="8" HEIGHT="2"
></TD>
        <TD></TD>
      </TR>
      <TR>
        <TD BGCOLOR="#333300"><SPACER TYPE="block" WIDTH="1" HEIGHT="1"></
TD>
        <TD BGCOLOR="#000000" VALIGN="center" ALIGN="center"><SMALL><FONT
COLOR="#ffffff"><B>
          16 Feb 2000
        </B></FONT></SMALL></TD>
        <TD BGCOLOR="#cccc99" VALIGN="center"><SPACER TYPE="block" WIDTH="
8" HEIGHT="1"></TD>
        <TD></TD>
        <TD><SMALL>
          Logged in as blholmes
```



```
mmerce3/ExecMacro/yourinfo.d2w/report%3Forder_rn%3D">
  <IMG SRC="https://www.borders.com/web_images/2/section-nav/commer
ce/1/log.gif" ALT="Log Out" WIDTH="145" HEIGHT="22" BORDER="0">
  </A>
```

```
<!-- LOGGEDIN-STATUS -->
<!-- begin name ID at left -->
<TABLE BORDER="0" WIDTH="145" CELLPADDING="0" CELLSPACING="0">
  <TR>
    <TD COLSPAN="3" HEIGHT="2"><SPACER TYPE="block" WIDTH="1" HEIGHT="2"></TD>
  </TR>
  <TR>
    <TD WIDTH="7"><SPACER TYPE="block" WIDTH="7" HEIGHT="1"></TD>
    <TD WIDTH="127">
      <SMALL>
        <B>Logged in as: <FONT COLOR="#cc0000">blholmes</FONT></B>

      <BR></SMALL>
    </TD>
    <TD WIDTH="11"><SPACER TYPE="block" WIDTH="11" HEIGHT="1"></TD>
  </TR>
  <TR>
    <TD COLSPAN="3" HEIGHT="4"><SPACER TYPE="block" WIDTH="1" HEIGHT="4"></TD>
  </TR>
  <TR>
    <TD COLSPAN="2" HEIGHT="1" BGCOLOR="#333300"><SPACER TYPE="block" WIDTH="1
HEIGHT="1"></TD>
    <TD WIDTH="11"></TD>
  </TR>
  <TR>
    <TD COLSPAN="3" HEIGHT="4"><SPACER TYPE="block" WIDTH="1" HEIGHT="4"></TD>
  </TR>
</TABLE>
<!-- end name ID at left -->
```

```
<!-- begin shopping bag status blurb at left -->
<TABLE BORDER="0" WIDTH="145" CELLPADDING="0" CELLSPACING="0">
  <TR>
    <TD WIDTH="7"><SPACER TYPE="block" WIDTH="7" HEIGHT="1"></TD>
    <TD WIDTH="127">
      <SMALL>
        You have <B>1 item</B> in your shopping bag for a subtotal of <FON
T COLOR="#cc0000"><B>$&nbsp;32.50</B></FONT>.
      <BR></SMALL>
    </TD>
    <TD WIDTH="11"><SPACER TYPE="block" WIDTH="11" HEIGHT="1"></TD>
  </TR>
  <TR>
    <TD COLSPAN="3" HEIGHT="4"><SPACER TYPE="block" WIDTH="1" HEIGHT="4"
></TD>
  </TR>
  <TR>
    <TD COLSPAN="2" HEIGHT="1" BGCOLOR="#333300"><SPACER TYPE="block" WI
```

```
DTH="1" HEIGHT="1"></TD>
      <TD WIDTH="11"></TD>
    </TR>
    <TR>
      <TD COLSPAN="3" HEIGHT="4"><SPACER TYPE="block" WIDTH="1" HEIGHT="4"
></TD>
    </TR>
  </TABLE>
<!-- end shopping bag status blurb at left -->
```

```
<!-- LEFT-CUST-SERV -->
<!-- begin name ID at left -->

<!-- end name ID at left -->
```

```
<!-- COLUMN-RIGHT-START -->
</TD>
<TD WIDTH="8" BGCOLOR="#ffffcc"><SPACER TYPE="block" WIDTH="8" HEIGHT="1"></TD>
```

>
<TD WIDTH="8"><SPACER TYPE="block" WIDTH="8" HEIGHT="1"></TD>
<TD VALIGN="top">

<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0" WIDTH="100%">
 <TR VALIGN="bottom">
 <TD ROWSPAN="5" WIDTH="406"></TD>
 >
 <TD WIDTH="100%" HEIGHT="34"></TD>
 </TR>
 <TR>
 <TD HEIGHT="1" BGCOLOR="#333333"><SPACER TYPE="block" WIDTH="1" HEIGHT="1">
 ></TD>
 </TR>
 <TR>
 <TD HEIGHT="3"></TD>
 </TR>
 <TR>
 <TD HEIGHT="1" BGCOLOR="#333333"><SPACER TYPE="block" WIDTH="1" HEIGHT="1">
 ></TD>
 </TR>
</TABLE>
Links to your account information are included below. Changes to your Address Book will not be saved unless you are logged in.

 <P></P>

 <TABLE BORDER="0" CELLPADDING="2" CELLSPACING="0" BGCOLOR="#ffffff" WIDTH="98%">
 <TR BGCOLOR="#CCCC99">
 <TD>
 Your Profile Information:
 </TD>
 </TR>
 </TABLE>

 <P></P>
 <TABLE WIDTH="98%" BORDER="0" CELLSPACING="0" CELLPADDING="4">
 <TR>
 <TD>
 <P>Order Status:

 To check the status of an order.</P>
 <P>Address Book:

 To view, edit, add or delete any shipping addresses in your Address Book.</P>
 >
 <P>Quick Checkout Profile:

 To create your Quick Checkout Profile.</P>

 <P>Your Profile Information:

 To view or update your profile username or password,
 email or address.</P>
 </TD>

```
</TR>
</TABLE>
<HR SIZE="1" WIDTH="98%" ALIGN="LEFT" NOSHADE>

<!-- COLUMN-BOTH-END -->

        </TD>
        </TR>
        </TABLE>
        </TD>
        </TR>
</TABLE>

<!-- COLUMN-LEFT-START -->

<TABLE BORDER="0" WIDTH="100%" CELSPACING="0" CELLPADDING="0">
  <TR>
    <TD BGCOLOR="#ffffff">
      <TABLE BORDER="0" CELSPACING="0" CELLPADDING="0">
        <TR>
          <TD WIDTH="1" BGCOLOR="#333300"><SPACER TYPE="block" WIDTH="1" HEI
GHT="1"></TD>
          <TD WIDTH="145" BGCOLOR="#cccc99" VALIGN="top">

<!-- BACK-TO-TOP -->

          <!-- Start back-to-top left side -->
          <TABLE WIDTH="145" BORDER="0" CELSPACING="0" CELLPADDING="2">
            <TR>
              <TD WIDTH="145">
                <SMALL><P>&nbsp;<A HREF="/borders/https://www.borders.com/cgi-bi
n/ncommerce3/ExecMacro/yourinfo.d2w/report?Krypto=UgWYjhAe%2FCd7qjrSZo4h3ZVf2d4L
mkPT1SbHgQr04RgWHz1gLPZ5idi5mLjcG95p0LQsq5PMnQyunlgrDpBaqf%2FBdHtb7GvQgr04zeJq9D
k2iGWpYKV4mOntEgTqM0S5vChXKAtX4KlTVjYpZaUgcTWIAD7Z0Lo%2F2%2BJeL40SAnjmrP7txKcBTN
dg7cFiHEJca16dJYPBCcrZR6HuECuc%2BWIB5q98ktqcLD0Ve%2FI9Fql5Sb%2FyRAwl74imWnZ6reIy
Afz6BFpouBrXuBSVonclVlqeQsRyPrCO8kbM4nxUtSw%3D#top">Back to top</A></P></SMALL>
              </TD>
            </TR>
          </TABLE>
          <!-- End GC left side -->

<!-- COLUMN-RIGHT-START -->
          </TD>
          <TD WIDTH="8" BGCOLOR="#ffffcc"><SPACER TYPE="block" WIDTH="8" HEI
GHT="1"></TD>
          <TD WIDTH="8"><SPACER TYPE="block" WIDTH="8" HEIGHT="1"></TD>
          <TD VALIGN="top">

<!-- GLOBAL-FOOTER -->

          <SMALL>
          <P>
            <A HREF="http://proxy.luciditygifts.com/borders/http://www.borde
rs.com/">Home</A> |
            <A HREF="http://proxy.luciditygifts.com/borders/http://www.borde
rs.com/sections/section_10000.html">Books</A> |
            <A HREF="http://proxy.luciditygifts.com/borders/http://www.borde
rs.com/sections/section_20000.html">Music</A> |
            <A HREF="http://proxy.luciditygifts.com/borders/http://www.borde
rs.com/sections/section_30000.html">Video</A> |
```

```
<A HREF="http://proxy.luciditygifts.com/borders/http://www.borde
rs.com/netcafe/index.html">NetCafe</A> |
<A HREF="http://proxy.luciditygifts.com/borders/http://www.borde
rs.com/sections/section_40000.html">Children's</A> |
<A HREF="http://proxy.luciditygifts.com/borders/http://www.borde
rs.com/sections/section_50000.html">Computer Books</A><BR>

<A HREF="/borders/https://www.borders.com/cgi-bin/ncommerce/Exec
Macro/shopcart.d2w/report">Shopping Bag</A> |
<A HREF="http://proxy.luciditygifts.com/borders/http://search.bo
rders.com/cgi-bin/db2www/search/search.d2w/BookKeyword">Expert Searches</A> |
<A HREF="http://proxy.luciditygifts.com/borders/http://go.border
s.com/help/customer-service.xcv">Customer Service</A> |
<A HREF="/borders/https://www.borders.com/cgi-bin/ncommerce/Regi
sterForm">Your Account</A>
<BR>
<A HREF="http://proxy.luciditygifts.com/borders/http://www.borde
rsstores.com/">Borders Stores</A> |
<A HREF="http://proxy.luciditygifts.com/borders/http://go.border
s.com/about.xcv">About Borders.com</A> |
<A HREF="http://proxy.luciditygifts.com/borders/http://go.border
s.com/disclaimers.xcv">Copyright &copy; 1998-2000</A>
</P>
<P>
Borders Online, Inc. is the authorized licensee of the Borders.c
om trademark and<BR>
domain name and the Borders NetCafe trademark. All rights reserv
ed.

</P>
</SMALL>

!-- COLUMN-BOTH-END -->

</TD>
</TR>
</TABLE>
</TD>
</TR>
</TABLE>

!-- END-HTML -->
</BODY>
</HTML>

!--
Socket time for this request: -0.0800
Server time for this request: 0.0900
-->
```

```
<%args>
$url
</%args>
<%init>
$r->send_cgi_header("Location: $url\n\n");
</%init>
```


<%INIT>

```
my ($prepend) = "";

if (grep(/^https:/, $url)) {
    $prepend = "https://sproxy.luciditygifts.com";
}
if (grep(/^http:/, $url)) {
    $prepend = "http://sproxy.luciditygifts.com";
}
$url = url($url, $base)->abs;
if ($type eq 'area' || $type eq 'a') {
    if (!grep(/javascript/i, $url)) {
        $url = "$prepend/borders/$url";
    }
}
if ($type eq 'form') {
    $url = "$prepend/borders/$url";
}
return $pre . $quote . $url . $quote;
```

</%INIT>

<%ARGS>

```
$pre=>undef
$type=>undef
$quote=>undef
$url=>undef
$base=>undef
/%ARGS>
```

```
<%perl_doc>
this component is used to send a return payment request to CyberCash CashRegister
</%perl_doc>
<%args>
$order_id
$amount
</%args>
<%once>
use CyberC::CCMckLib3_2;
use CyberC::CCMckDirectLib3_2;
use CyberC::CCMckErrno3_2;

# Initialize the Global Configuration array in CCMckLib3_2
# JW - change setting of $ConfigFile variable based on where it will eventually
reside and confirm giftcat names
my $ConfigFile;
if($ARGS{giftcat} eq 'charities'){
    $ConfigFile = '/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/con
f/merchant_conf';
}
elsif ($ARGS{giftcat} eq 'giftcert'){
    $ConfigFile = '/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/con
f/shine_conf';
}
my $status = CyberC::CCMckLib3_2::InitConfig($ConfigFile);
</%once>
<%init>
my (%result) = CyberC::CCMckDirectLib3_2::SendCC2_1Server('return', %ARGS);
return (\%return);
</%init>
```

```
<%init>
$otburl = "http://10.20.1.53/process/return_int_vars?u_id=$u_id";
my $req = new HTTP::Request 'GET', $otburl;
my $otb_var;

my $res = $ua->request($req);
if ($res->code eq '200') {
    @pairs = split(/\&/, $res->content);
    foreach $pair (@pairs) {
        ($name, $value) = split(/\=/, $pair);
        $otb_var{$name} = $value;
    }
    return $otb_var{'balance'};
} else {
    return "HTTP/1.0 200 OK\nContent-type: text/html\n\n<h2>fatal error 3561100-
012: unable to reach return_int_vars server.\n";
    exit;
}
</%init>
<%ARGS>
$u_id=>undef
</%ARGS>
```

```
<%init>
my($fullDate);
$dest_address =~ s/\s+/, /g;
chomp($fullDate = `date`);
# Do we want to append/prepend to text at all
# Create the mail command
my($mail_cmd) = "To: $dest_address\nFrom: $reply_to\nSubject: $subject\n";
$mail_cmd .= "Reply-to:$reply_to\n";
if ($bounce_to) {
    $mail_cmd .= "Bounce-to:$bounce_to\n";
}
$mail_cmd .= "Error:$fullDate\n$text\n";
open (SENDMAIL, "| /usr/lib/sendmail -t");
print SENDMAIL $mail_cmd;
close(SENDMAIL);

</%init>
<%ARGS>
$dest_address=>'fatalerror@ligifts.com'
$text=>'An Error occurred'
$subject=>'Error '.localtime
$reply_to=>'tech@ligifts.com'
$bounce_to=>'tech@ligifts.com'
</%ARGS>
```

```
%if ($ct eq 'text/html') {  
<table width=100% bgcolor="#000000"><tr><td><font color=white>Lucidity HTTP Pr  
oxy -- Your requests are currently being proxied.</font></td></tr></table>  
%}  
</%INIT>  
<%ARGS>  
$ct=>undef  
</%ARGS>
```

```
<%INIT>
mc_suppress_http_header(1);

my ($l_name, $l_gcn, $l_gce, $l_mm, $l_yyyy, $l_uid, $l_damt);
my $timestamp = time;
my $trail_pre;
if ($ARGS{'emethod'} ne '') {
    $ENV{'REQUEST_METHOD'} = "GET";
}

my $req;
$req = new HTTP::Request $ENV{'REQUEST_METHOD'}, $url;

# Some sites need a real looking referer.
$req->push_header("Referer", $url);
$req->push_header("Authorization", $ENV{'HTTP_AUTHORIZATION'});

# Proxy True Agent
$ua->agent($ENV{'REMOTE_AGENT'});
$ua->agent("Mozilla/4.7 [en] (X11; U; Linux 2.2.10 i586)");

my $res = $ua->request($req);

# Return the proper response code.
my $code = $res->code;
die ("bad return code:$code for $ENV{'REQUEST_METHOD'} $url") unless $code eq
200;

# Convert to absolute references
my $content = $res->content;
my $base = $res->base;
/%INIT>

<% $content %>

<%ARGS>
$url=>undef
$ua=>undef
/%ARGS>
```

```
<%INIT>
my $url = $ARGS{ucid_url};

my $replace_link = sub {
    my($pre, $type, $quote, $url, $base) = @_;
    my ($prepend) = "https://sproxy.luciditygifts.com";

    if (grep(/^https\/:/, $url)) {
        # $prepend = "https://sproxy.luciditygifts.com";
    }
    if (grep(/^http\/:/, $url)) {
        # $prepend = "http://proxy.luciditygifts.com";
    }
    my $u1 = URI::URL->new($url, $base);
    $url = $u1->abs;
    # $url = url($url, $base)->abs;
    if ($type eq 'area' || $type eq 'a') {
        # if (!grep(/javascript/i, $url)) {
        #     $url = "$prepend/bma/$url";
        # }
    }
    if ($type eq 'form') {
        # $url = "$prepend/bma/$url";
    }
    $url = "$prepend/bma/$url";

    return $pre . $quote . $url . $quote;
}

my $timestamp = time;
my $trail_pre;
if (grep(/^https\/:/, $url)) {
    $trail_pre = "https://strail.luciditygifts.com";
} else {
    $trail_pre = "http://trail.luciditygifts.com";
}
my $args = mc_comp( '/shared/utills/makeURI', %ARGS );

my $req;
if ($ENV{'REQUEST_METHOD'} eq 'POST') {
    $req = new HTTP::Request $ENV{'REQUEST_METHOD'}, "$url";
    $req->content_type('application/x-www-form-urlencoded');
    $req->content($args);
} elsif ($args) {
    $url = "$url?$args";
    $req = new HTTP::Request $ENV{'REQUEST_METHOD'}, $url;
} else {
    $req = new HTTP::Request $ENV{'REQUEST_METHOD'}, $url;
}

# Proxy True Agent
$ucid_ua->agent($ENV{'REMOTE_AGENT'});

my $res = $ucid_ua->request($req);

my $ct = $res->content_type;
$res->content_type($ct);
$res->headers_out->unset('Content-Type');
$res->headers_out->set('Content-Type'=>"$ct");
```

```

# Return the proper response code.
my $code = $res->code;
$r->headers_out->add('Code'=>"$code");

$r->send_http_header;

$m->abort($code) unless $code eq 200;

# Convert to absolute references
my $content = $res->content;
my $base = $res->base;
$content =~ s/
(
  <(img|a|body|area|frame|td|input|form|link|script)\b
  [^>]+
  \b(?:src|href|background)
  \s*=\s*
  )
  (?:( "[^"]*" ) | ( ' [^']* ' ) | ([^\s>]+) )
/
$replace_link->( $1, lc($2), $3||$5, HTML::Entities::decode($4||$6||$7), $base
) /giex;

my $total_purchase = 0;

# We do absolute paths and disable base hrefs
$content =~ s/base href="\s*/basehref="\s*/ig;

# Handle Meta Refresh **** FIX REGEX TO HANDLE ANY DELAY ***
$content =~ s/"0\;URL=\s*"0\;URL=\s*/bma//g;
$content =~ s/"3\;URL=\s*"3\;URL=\s*/bma//g;

/%INIT>
if ($r->content_type =~ /image/) {
  <% $content %>
} else {
  <% $content %>
  # Pixel Calls
  if (grep(/\<basehref=\s*"http:\/\/www.bluemountainarts.com\/\s*>/, $content)) {
    
  }
  if (grep(/\<TITLE>Borders.com - Standard Checkout<\/TITLE>/, $content)) {
    
  }
  if (grep(/\<TITLE>Borders.com - Your Receipt<\/TITLE>/, $content)) {
    if (grep(/BGCOLOR="\s*#FFFFCC\s*">\<B>Total<\/B>>\<\/TD>/, $content)) {
      my ($sa, $sb) = split(/BGCOLOR="\s*#FFFFCC\s*">\<B>Total<\/B>>\<\/TD>/, $content)
      \<TD ALIGN="\s*RIGHT\s*" width=70 BGCOLOR="\s*#FFFFCC\s*"
"><FONT SIZE="\s*2\s*">\<B>\$/, $content);
      ($total_purchase, $sb) = split(/\<\/B>/, $sb);
      $total_purchase =~ s/ //g;
    }
    
  }
}

```



```
%}  
<%ARGS>  
$lucid_ua=>undef  
</%ARGS>
```

-----BEGIN CERTIFICATE-----

MIIC9TCCA16gAwIBAgIDAKVeMA0GCSqGSIb3DQEBAUAMIHEMQswCQYDVQQGEwJa
QTEVMBMGA1UECBMMV2VzdGVybiBDYXB1MRIwEAYDVQQHEw1DYXB1IFRvd24xHTAb
BgNVBAoTFFRoYXZ0ZSBDb25zdWx0aW5nIGNjMSgwJgYDVQQLEx9DZXJ0aWZpY2F0
aW9uIFN1cnZpY2VzIERpdmlzaW9uMRkwFwYDVQQDExBUaGF3dGUgU2VydmlVYIENB
MSYwJAYJKoZIhvcNAQkBFhdzZXJ2ZXItY2VydHNAZGhhd3R1LmNvbTAeFw05OTEy
MDExMDIwNTdaFw0wMDEyMTQxMDIwNTdaMIGSMQswCQYDVQQGEwJVUzETMBEGA1UE
CBMKQ2FsaWZvcn5pYTEwMBMGA1UEBxMNU2FuIEZyYW5jaXNjbzEXMBUGA1UEChMO
THVjaWRpdHksIEluYy4xGjAYBgNVBAsTEWx1Y21kaXR5Z21mdHMuY29tMSEwHwYD
VQDExhzchJveHkubHVjaWRpdHlnaWZ0cy5jb20wgZ8wDQYJKoZIhvcNAQEBBQAD
gY0AMIGJAoGBAL4t0xyGgKDXhFwV+NMNAkUaI4k8w9TbCVIH84IZhyB0hyj3W3wN
090fG+K+nnoEkKSuf6Bw9Yu9ITU3HDTEf79WyoW0puY88p3zGwg4LH6gcQe0cTN9
IRBpgJsaNsJNpo3UYUCRDNAP5tt6ja/BVucl7dZcEpacgYrJUyXqJQJzAgMBAAGj
JTAjMBMGA1UdJQQMMAoGCCSGAQUFBwMBMAwGA1UdEwEB/wQCMAAwDQYJKoZIhvcN
AQEEBQADgYEAfnSisWvsbRRmrtPWj2cDKJoub0XPz/3OfiqMS9MLNIRlfVvr3OY
iONe6CQsFWMwSZHag+YM/jwS3iULogbaveXVNW18mbJmgkNV7EMBb5BnCV7Y5aEx
IkwlAmZrjfrJO17C7XE3yV8smkyITsO6UpexXKA0gcf+6bh2Ij+gnps=

-----END CERTIFICATE-----

-----BEGIN CERTIFICATE REQUEST-----

MIIB+TCCAWICAQAwgbgxCzAJBgNVBAYTA1VTMRMwEQYDVQQIEwpDYWxpZm9ybmlh
MRYwFAYDVQQHEw1TYW4gRnJhbmNpc2NvMRcwFQYDVQQKEw5MdWNPZG10eSwgSW5j
LjEaMBGGA1UECzMrbHVjaWRpdHlnaWZ0cy5jb20xITAfBgNVBAMTGHNwcm94eS5s
dWNPZG10eWdpZnRzLmNvbTEKMCIGCSqGS1b3DQEJARYVamFzb250QGNvbml5Y3Ru
ZXQuY29tMIGfMA0GCSqGSIb3DQEBQUAA4GNADCBiQKBgQC+LdMchoCg14RcFfjT
DQJFGiOJPMPU2wLSB/OCGYcgdIco91t8DdPdHxviyp56BJCkrm+gcPWLvSE1Nxxw0
3n+/VsGMNKbmPPKd8xsIOCx+oHEHtHEzfSEQaaibGjbCTaan1GFHEQzQD+bbeo2v
wVbnJe3WXBKWNIGKyVGF6iUCcwIDAQABoAAwDQYJKoZIhvcNAQEEBQADgYEAm9Fx
yNy9h0Cz7P91B0ZMumNAkveU2PD4kMgbplFMWw88seFDR/I64MmCEIFdSrZvLwXO
2IsU/CraYOvATAPEHNWmJ644NpHNrWcnquWksR9zxoVJ0f/4bJNP9yUeSS9VHpiV
KTbcZ0xUQW8a19UwJu19++h3Cul2ow2TlS8ifw4=

-----END CERTIFICATE REQUEST-----

```
<%INIT>
my @states = (
    { code => 'AL', name => 'Alabama' },
    { code => 'AK', name => 'Alaska' },
    { code => 'AZ', name => 'Arizona' },
    { code => 'AR', name => 'Arkansas' },
    { code => 'CA', name => 'California' },
    { code => 'CO', name => 'Colorado' },
    { code => 'CT', name => 'Connecticut' },
    { code => 'DE', name => 'Delaware' },
    { code => 'DC', name => 'District of Columbia (DC)' },
    { code => 'FL', name => 'Florida' },
    { code => 'GA', name => 'Georgia' },
    { code => 'HI', name => 'Hawaii' },
    { code => 'ID', name => 'Idaho' },
    { code => 'IL', name => 'Illinois' },
    { code => 'IN', name => 'Indiana' },
    { code => 'IA', name => 'Iowa' },
    { code => 'KS', name => 'Kansas' },
    { code => 'KY', name => 'Kentucky' },
    { code => 'LA', name => 'Louisiana' },
    { code => 'ME', name => 'Maine' },
    { code => 'MD', name => 'Maryland' },
    { code => 'MA', name => 'Massachusetts' },
    { code => 'MI', name => 'Michigan' },
    { code => 'MN', name => 'Minnesota' },
    { code => 'MS', name => 'Mississippi' },
    { code => 'MO', name => 'Missouri' },
    { code => 'MT', name => 'Montana' },
    { code => 'NC', name => 'N. Carolina' },
    { code => 'ND', name => 'N. Dakota' },
    { code => 'NE', name => 'Nebraska' },
    { code => 'NV', name => 'Nevada' },
    { code => 'NH', name => 'New Hampshire' },
    { code => 'NJ', name => 'New Jersey' },
    { code => 'NM', name => 'New Mexico' },
    { code => 'NY', name => 'New York' },
    { code => 'OH', name => 'Ohio' },
    { code => 'OK', name => 'Oklahoma' },
    { code => 'OR', name => 'Oregon' },
    { code => 'PA', name => 'Pennsylvania' },
    { code => 'RI', name => 'Rhode Island' },
    { code => 'SC', name => 'South Carolina' },
    { code => 'SD', name => 'South Dakota' },
    { code => 'TN', name => 'Tennessee' },
    { code => 'TX', name => 'Texas' },
    { code => 'UT', name => 'Utah' },
    { code => 'VT', name => 'Vermont' },
    { code => 'VA', name => 'Virginia' },
    { code => 'WA', name => 'Washington' },
    { code => 'WV', name => 'West Virginia' },
    { code => 'WI', name => 'Wisconsin' },
    { code => 'WY', name => 'Wyoming' },
);

return( @states );

</%INIT>
```

```
<%init>
return unpack('H*', $old); # Can return huge string
</%init>
<%args>
$old
</%args>
```

```
#!/usr/bin/perl

use HTML::Mason;
use strict;

my $outbuf;
my $parser = new HTML::Mason::Parser;
my $interp = new HTML::Mason::Interp( parser=>$parser,
                                       comp_root=>'/home/www/comps/shared/utils',
                                       data_dir=>'/home/www/mason/shared/utils',
                                       out_method=>\$outbuf );

my $retval = $interp->exec( '/unique_id.new' );

open( F, "mason.out" );
print F $outbuf;
close(F);
print "return value of component was: $retval\n";
```

```
<%ONCE>
use LWP::UserAgent;
use HTTP::Request;
</%ONCE>
<%INIT>

my $ua = new LWP::UserAgent;
$ua->agent("Mozilla/4.7 [en] (Lucidity)");
$ua->timeout(20);

my $req = new HTTP::Request 'GET', "https://www.borders.com";
my $bma_res = $ua->request($req);
</%INIT>
<% $bma_res->code %>
<% $bma_res->content %>
```

```
#!/usr/local/bin/perl

use LWP::UserAgent;
use HTTP::Request;

my $ua = new LWP::UserAgent;
$ua->agent("Mozilla/4.7 [en] (Lucidity)");
$ua->timeout(20);

my $req = new HTTP::Request 'GET', "https://www.borders.com";
my $bma_res = $ua->request($req);
print $bma_res->code;
print $bma_res->content;
```



```
<%doc>
Process to generate unique UUIDs. This algorithm will continue to work properly u
ntil the year 2587 :)
Based on GMT time and the process ID number ($$)
</%doc>
<%init>
my( $ID,$hex );
my( $sec,$min,$hour,$mday,$mon,$year,$yday,$isdst ) = gmtime(time());
my $rand = int(rand(10));

# create ID based on date information and process number
# $yday = day of the year (0-365); $yday+1 to get the more familiar 1-366
# $year is years since 1900 (3 digits)
# $msec = seconds since midnight (5 digits, since it's at most 86400)

my $msec = ($hour*3600)+($min*60)+$sec;
my $num = sprintf( "%03d%03d%05d", $year,$yday+1,$msec );

# Convert to hex to compress number from 11 to 9 characters while keeping it uni
que
# (hex function produces overflow so this is done explicitly)

my $test = sprintf( "%9x",$num );

my( $i );
my $temp = $num;
while( $temp >= 16 ) {
    $temp = $temp/16;
    $i++;
}
while( $i>=0 ) {
    my $exp = 16**$i;
    my $value = int($num/$exp);
    if( $value == 10 ) {
        $value = 'A';
    } elsif( $value == 11 ) {
        $value = 'B';
    } elsif( $value == 12 ) {
        $value = 'C';
    } elsif( $value == 13 ) {
        $value = 'D';
    } elsif( $value == 14 ) {
        $value = 'E';
    } elsif( $value == 15 ) {
        $value = 'F';
    }
    $hex = $hex . $value;
    $num = $num - ($exp*$value);
    $i--;
}

while( length($hex) < 9 ) {
    $hex = '0' . $hex;
}

# Add PID and a random number
$ID = sprintf( "%09d%05d%01d",$hex,$$, $rand );
```

```
return $ID;
</%init>
```

[illegible]

```
<%doc>
borrowed code from CyberCash to generate unique id's
based on GMT time and the process ID number ($$)
</%doc>
<%init>
my($ID);
my($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = gmtime(time());

# create ID based on date, time, and process number
# $mon is the month index where Jan=0 and Dec=11, so we use
# $mon+1 to get the more familiar Jan=1 and Dec=12
# $year is years since 1900: we want only two digits

$ID = sprintf("%02d%02d%02d%02d%05d", $year % 100, $mon+1,$mday,$hour,$min,$
$);

return $ID;
</%init>
```

```
<%doc>
borrowed code from CyberCash to generate unique id's
based on GMT time and the process ID number ($$)
</%doc>
<%init>
my($ID);
my($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = gmtime(time());

# create ID based on date, time, and process number
# $mon is the month index where Jan=0 and Dec=11, so we use
# $mon+1 to get the more familiar Jan=1 and Dec=12
# $year is years since 1900: we want only two digits

$ID = sprintf("%02d%02d%02d%02d%05d", $year % 100, $mon+1,$mday,$hour,$min,$
$);

return $ID;
</%init>
```

```
%foreach my $key (keys %ARGS) {  
    <% $key %>=<% $ARGS{$key} %><BR>  
}%}
```

20000509182205

```
<%args>
$old
$fname
$offset
</%args>
<%init>
my ($key);

open (KEY, $fname) or die "Cannot open $fname: $!";
seek(KEY, $offset,0);
read(KEY, $key, 256);
close KEY;

return substr ($old ^ $key, 0, length $old); # Assumes length $old <= length $key
Y
#return substr ($old ^ $key, 0, (length $old < length $key) ? length $old : leng
th $key); # More robust
</%init>
```

```
<%PERL_ARGS>
$first_year=>'
$show_many=>'20'
$selected=>'
$name=>'year'
</%PERL_ARGS>
<%PERL_INIT>
$first_year = 1999;      # Replace with call to get year...

</%PERL_INIT>

<SELECT NAME="<%%$name%>">
  <OPTION VALUE=" ">Year
  % for( my $year=$first_year ; $year(<$first_year+$show_many) ; $year++ ) {
  %   if( $selected == $year ) {
  %     <OPTION VALUE="<%%$year%>" SELECTED><%%$year%>
  %   } else {
  %     <OPTION VALUE="<%%$year%>"><%%$year%>
  %   }
  % }
</SELECT>
```

fund

Copy 1 of 3

2025 RELEASE UNDER E.O. 14176


```

% if(defined($process)){
%   (%missing) = mc_comp('add_validate', %ARGS);
%   unless ($missing{status}) {
%     if ($process eq "CONFIRM") {
%       <& add_send, %ARGS &>
%       $called = "add_send";
%     } elsif ($process eq "INFO") {
%       $ARGS{process} = "CONFIRM";
%       $called = "add_confirm";
%       <& add_template, %ARGS, top=>1&>
%       <& add_confirm, %ARGS &>
%       <& add_template, %ARGS, bottom=>1&>
%     }
%   } else {
%     $called = "add_info: $process";
%     <& add_template, title=>'Add Funds to Your Bluemountain Gold', %ARGS, t
op=>1&>
%     <& add_info, missing=>\%missing, %ARGS &>
%     <& add_template, %ARGS, bottom=>1&>
%   }
% } else {
%   $called = "add_info no process";
%   <& add_template, title=>'Add Funds to Your Bluemountain Gold', %ARGS, top=>
1&>
%   <& add_info, %ARGS &>
%   <& add_template, %ARGS, bottom=>1&>
% }

<%INIT>
$ARGS{giftcat}      = 'giftcert';
$ARGS{card_number}  = ~ s/[^d]+//g;
$ARGS{order_id}     = mc_comp('/shared/utls/unique_id') unless $ARGS{order_id}
};
$ARGS{u_id}         = $ARGS{uid} unless $ARGS{u_id};
unless ($ARGS{balance}) {
  my (%data)        = mc_comp('user_lookup', u_id=>$ARGS{u_id});
  $ARGS{balance}    = $data{balance};
}
if ($ARGS{damt} && ! $ARGS{amount}) {
  $ARGS{amount}     = sprintf '%.2f', $ARGS{damt}-$ARGS{balance};
}
$ARGS{amount}       = ~ s/[^d\.]+//g;
my %missing;
my $called;

</%INIT>
<%ARGS>
$process=>undef
</%ARGS>

<!--
Called <% $called %>
% foreach my $key (sort keys(%ARGS)) {
%   my $value = $ARGS{$key};
%   Key: <%%$key%> Value: <%%$value%>
% }
% foreach my $key (sort keys(%missing)) {
%   my $value = $missing{$key};
%   Missing Key: <%%$key%> Value: <%%$value%>
% }

```

-->

```

% if(defined($process)){
%   (%missing) = mc_comp('add_validate', %ARGS);
%   unless ($missing{status}) {
%     if ($process eq "CONFIRM") {
%       <& add_send, %ARGS >
%       $called = "add_send";
%     } elsif ($process eq "INFO") {
%       $ARGS{process} = "CONFIRM";
%       $called = "add_confirm";
%       <& add_template, %ARGS, top=>1&>
%       <& add_confirm, %ARGS >
%       <& add_template, %ARGS, bottom=>1&>
%     }
%   } else {
%     $called = "add_info: $process";
%     <& add_template, title=>'Add Funds to Your Bluemountain Gold', %ARGS, t
op=>1&>
%     <& add_info, missing=>\%missing, %ARGS >
%     <& add_template, %ARGS, bottom=>1&>
%   }
% } else {
%   $called = "add_info no process";
%   <& add_template, title=>'Add Funds to Your Bluemountain Gold', %ARGS, top=>
1&>
%   <& add_info, %ARGS >
%   <& add_template, %ARGS, bottom=>1&>
% }
<%INIT>
$ARGS{giftcat}      = 'giftcert';
$ARGS{card_number}  =~ s/[^\\d]+//g;
$ARGS{order_id}     = mc_comp('/shared/utls/unique_id') unless $ARGS{order_id}
};
$ARGS{u_id}         = $ARGS{uid} unless $ARGS{u_id};
unless ($ARGS{balance}) {
  my (%data)        = mc_comp('user_lookup', u_id=>$ARGS{u_id});
  $ARGS{balance}    = $data{balance};
}
if ($ARGS{damt} && ! $ARGS{amount}) {
  $ARGS{amount}     = sprintf '%.2f', $ARGS{damt}-$ARGS{balance};
}
$ARGS{amount}       =~ s/[^\\d\\.]+//g;
my %missing;
my $called;
</%INIT>
<%ARGS>
$process=>undef
</%ARGS>

<!--
Called <% $called %>
% foreach my $key (sort keys(%ARGS)) {
%   my $value = $ARGS{$key};
%   Key: <%%$key%> Value: <%%$value%>
% }
% foreach my $key (sort keys(%missing)) {
%   my $value = $missing{$key};
%   Missing Key: <%%$key%> Value: <%%$value%>
% }

```

-->

Year	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099
1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	

4

```

<TABLE BORDER=0 BGCOLOR="#66CCFF" CELSPACING=0 CELLPADDING=4 WIDTH=700><!FFFFCC
>
<TR>
<TD VALIGN=middle BGCOLOR="#66CCFF" ALIGN=center valign=top>
<FONT SIZE=5 COLOR="#000000">
    To proceed, <b>click on submit!</b>
    You will be e-mailed a receipt,
%if ($store) {
    and you will be returned to the store complete your check-out.
}%
    Enjoy your Blue Gold!

%# Evil Here now:
% foreach my $key (keys %ARGS){
%   if ($key eq 'process') {
%       <INPUT TYPE="hidden" name="process" value="CONFIRM">
%   } else {
%       <INPUT TYPE="hidden" name="<%$key%>" value="<%%$ARGS{$key}%>">
%   }
% }
</font>
<p>
<b> <INPUT TYPE=image src="/images/giftcard/submitbutt.gif" Name="submit"></b><b>
r>
<font size=3>It may take up to a minute to process your transaction.</font>
<center>or<br>
<a href="javascript:history.go(-1)">go back and edit your order</a>
<p>&nbsp;<p></center>
</TD>
</TR>
<TR>
<TD align=left bgcolor=66CCFF>&nbsp;<br>
<p>&nbsp;<p></center>
</FORM>

<font size=2><i>Terms of Service:<br>
Blue Mountain Gold is brought to you by Bluemountain in partnership with Lucidit
y Gifts.
Once you submit, the recipient will be sent an e-mail notification, and will rec
eive a
subsequent reminder if he/she forgets
to view the card. You will receive an e-mail receipt. In 3 weeks if the recipien
t
has not yet viewed the card, you will receive an alert in your e-mail.
If the recipient does not pick up his/her gift certificate, you may request a re
fund.
<b>
You will be able to view and use your certificate for 6 months.
</b> <br>
<center><font size=1><i>If you have questions, you may e-mail <a href=mailto:Sup
port@BluemountainGifts.com>
Support@BluemountainGifts.com</a>.</center>
</b>
</font>
</TABLE>

```

```
<%ARGS>  
$store=>undef  
</%ARGS>
```

[illegible]

```
</TR>
<TR><TD COLSPAN=2><B>Credit Card Information</B></TD></TR>
<TR>
  <TD COLSPAN=2 ALIGN=middle>
    <TABLE BORDER=0 CELLPADDING=3 CELLSPACING=0>
      <TR>
        <TD WIDTH=20></TD>
        <TD><% $missing{'card_type'} %>Credit Card Type:</TD>
        <TD COLSPAN=7>
          <%mc_comp( '/shared/interface/dropdown/credit_card', name='card_type',
e', selected=>$card_type, giftcat=>$ARGS{'giftcat'} )%>
        </TD>
      </TR>
      <TR>
        <TD WIDTH=20></TD>
        <TD><% $missing{'card_number'} %>Credit Card number:</TD>
        <TD COLSPAN=7>
          <INPUT TYPE=text SIZE=20 NAME="card_number" VALUE="<%$card_number%"
MAXLENGTH=20>
        </TD>
      </TR>
      <TR>
        <TD WIDTH=20></TD>
        <TD><% $missing{'card_exp_month'} %><% $missing{'card_exp_year'} %>Exp
iration date:</TD>
        <TD COLSPAN=7>
          <& /shared/interface/dropdown/month, name='card_exp_month',selected
=>$card_exp_month &>
          <& /shared/interface/dropdown/year, name='card_exp_year', selected
=>$card_exp_year &>
        </TD>
      </TR>
      <TR>
        <TD WIDTH=20></TD>
        <TD COLSPAN=8>
          <B>Billing address</B> - Please supply the billing address for this
credit card:
        </TD>
      </TR>
      <TR>
        <TD WIDTH=20></TD>
        <TD><% $missing{'company'} %>Company:</TD>
        <TD COLSPAN=7><INPUT NAME="company" SIZE=30 MAXLENGTH=50 VALUE="<%$com
pany%"></TD>
      </TR>
      <TR>
        <TD WIDTH=20></TD>
        <TD><% $missing{'card_street'} %>Street:</TD>
        <TD COLSPAN=4><INPUT NAME="card_street" SIZE=30 MAXLENGTH=40 VALUE="<%
$card_street%"></TD>
```



```
<TD WIDTH=10></TD>

<TD><% $missing{'card_apt'} %>Apt #:</TD>
<TD><INPUT NAME="card_apt" SIZE=6 MAXLENGTH=5 VALUE="<%$card_apt%>"></
TD>
</TR>

<TR>
<TD WIDTH=20></TD>

<TD><% $missing{'card_city'} %>City:</TD>
<TD><INPUT NAME="card_city" SIZE=20 MAXLENGTH=20 VALUE="<%$card_city%>
"></TD>

<TD WIDTH=10></TD>

<TD><% $missing{'card_state'} %>State:</TD>
<TD><INPUT NAME="card_state" SIZE=3 MAXLENGTH=2 VALUE="<%$card_state%>
"></TD>

<TD WIDTH=10></TD>

<TD><% $missing{'card_zipcode'} %>Zip Code:</TD>
<TD><INPUT NAME="card_zipcode" SIZE=10 MAXLENGTH=10 VALUE="<%$card_zip
code%>"></TD>
</TR>

<TR>
<TD WIDTH=20></TD>
<TD><% $missing{'card_country'} %>Country:</TD>
<TD COLSPAN=7><INPUT NAME="card_country" SIZE=16 MAXLENGTH=32 VALUE="<
%($card_country ? "$card_country" : 'USA')%>"></TD>
</TR>

<TR>
<TD WIDTH=20></TD>
<TD COLSPAN=8><B>Phone number</B> - associated with the billing addres
s</TD>
</TR>

<TR>
<TD WIDTH=20></TD>

<TD COLSPAN=9>
<% $missing{'area_code'} %>Area:
<INPUT NAME="area_code" SIZE=4 MAXLENGTH=3 VALUE="<%$area_code%>">
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
<% $missing{'phone'} %>Phone #:
<INPUT NAME="phone" SIZE=10 MAXLENGTH=8 VALUE="<%$phone%>">
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
<% $missing{'ext'} %>Ext:
<INPUT NAME="ext" SIZE=5 MAXLENGTH=6 VALUE="<%$ext%>">
</TR>
</TABLE>

</TD>
</TR>

<TR>
<TD COLSPAN=8>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
</TD>
```

```
</TABLE><center><font size=5 face=arial,helvetica><b>
<INPUT TYPE=image src=<% $image_base %>/continuebuttc.gif border=0 NAME="submit"
>
</b></font><br>
```

continuing will bring you to a purchase con
firmation page

[privacy and security statement](http://www.luciditygifts.com/giftcard/<%$ARGS{'type'}%>/help/policies.html)

<%INIT>

```
my $star = "<FONT COLOR=red>*</FONT>";
```

```
#----- Translate BMA variables into Lucidity variables if supplied ---
-----#
```

```
$to_email    = $EF    unless $to_email;
$name       = $NF    unless $to_name;
$id         = $uid   unless $id;
```

```
$card_first_name = $to_name unless $card_first_name;
my $image_base = "https://fund.luciditygifts.com/images/giftcard";
</%INIT>
```

[illegible]

11

```

<HTML>
<HEAD>
  <TITLE>Blue Mountain Gold Page</TITLE>
  %# <META http-equiv="refresh" content="15, url=http://www.luciditygifts.com/gif
tcard/balance?LOC=<% $LOC %>&type=<%$type%>">
</HEAD>

<BODY BGCOLOR="#ffffff" BACKGROUND="/images/charity/clouds_bg.gif">
<center>
<TABLE bgcolor=#ffffff border="0" CELLPADDING="0" CELLSPACING="0" WIDTH="700">
<tr><td align=right><img src=/images/giftcard/barrelscriptxt.gif>
</td><td align=center><font size=4>Congratulations! Your Blue Mountain Gold<font
size=2> (tm)<font size=4> gift certificate has been updated! <% $amount %> has
beed added to your account<font></td></tr></table>
<table width=100% bgcolor=#99ccff cellpadding=0 cellspacing=0>
<tr>
<td align=center>
You may continue shopping. Your current total is $<% $balance %><BR>
You will be e-mailed a receipt. Thank you.<br>
<font size=2><i>If you have questions about Blue Mountain Gold, please visit
the <a href=http://www.luciditygifts.com/charity/helpcenter/>
Blue Mountain Gold support page</a>, or e-mail <a href=mailto:info@lucidityinc.c
om>
info@lucidityinc.com</a></i>
</td></tr></table>

%if ($returnurl) {
%$returnurl = uri_escape($returnurl, "^A-Za-z0-9");
%#<P><A HREF="http://www.luciditygifts.com/giftcard/mall?LOC=<% $LOC %>&returnur
l=<% $returnurl %>" TARGET="mw">Continue Shopping</A><P>
%#<P><A HREF="<% $returnurl %>" TARGET='mw'>Continue Shopping</A><P>
%}
<P><A HREF="javascript:self.close()">Close</A><P>

%!--
% foreach my $key (keys %ARGS) {
  <% $key %> => <% $ARGS{$key} %><BR>
% }
%-->
</BODY>
</HTML>
<%ONCE>
use URI::Escape;
</%ONCE>
<%INIT>
$balance = sprintf "%.2f", $balance+$amount;
</%INIT>
<%PERL_ARGS>
$LOC=>'
$u_id=>'
$amount=>'
$balance=>'
$order_id=>'
$type=>'kids'
$returnurl=>undef
</%PERL_ARGS>

```

<%doc>

Make the call to charge and return data

Will display a frame set on success.

an error page with key/val pairs for any problems

</%doc>

<%once>

use URI::Escape;

</%once>

<%init>

my(%data, \$info_string);

#----- Call to app server to charge the card -----#

mc_comp('/shared/utills/info_string', %ARGS, STORE=>\\$info_string);

my \$giftcard = "10.20.1.53";

my \$url = "http://\$giftcard/process/charge?\$info_string";

#----- Get the LOC based on the u_id. Need these to redirect back to m
all -----#

my \$LOC = mc_comp('get_LOC', u_id=>\$u_id);

\$LOC =~ s/\s+//g;

#----- Set various URLs -----#

my \$return = mc_comp('/shared/http/lwp_call', url=>\$url, timeout=>90, retry=>1
, data=>\%data);

#unless (\$returnurl) {

my \$trailp = "10.20.1.54";

my \$trail_ret = mc_comp('/shared/http/lwp_call',

url=>"http://\$trailp/push/get_last_request?u_id=\$u_id", timeout=>10, ret

#ry=>3,

data=>\\$returnurl);

chomp (\$returnurl);

#}

</%init>

%if (\$return != 200 || \$data{success} == 0) {

<& error_template, %ARGS, %data, top=>1&>

<& error, %ARGS, %data, return=>\$return, url=>\$url, error_id=>\$info_string

<& error_template, bottom=>1&>

return 1;

%} elsif (\$data{success} == 3) {

% my (%missing);

% \$missing{status} = 1;

% \$missing{card_number} = "Your Card was declined, please try
another card.";

<& add_template, %ARGS, %data, top=>1&>

<& add_info, missing=>\%missing, %ARGS &>

<& add_template, %ARGS, %data, bottom=>1&>

% return 1;

%} elsif (\$data{success} == 2) {

% \$ARGS{queue} = 1;

%} elsif (\$data{success} == 4) {

% \$ARGS{duplicate} = 1;

%}

```
<& 'add_waiting', %ARGS, LOC=>$LOC &>
%# <& 'add_receipt', %ARGS, LOC=>$LOC &>
%# <%$refresh_url%>
```

```
<%ARGS>
$returnurl=>undef
$type=>'kids'
$u_id=>undef
</%ARGS>
```

14

```
<%doc>
Make the call to charge and return data
Will display a frame set on success.
    an error page with key/val pairs for any problems
</%doc>
<%init>
my( %data, $info_string );

#----- Call to app server to charge the card -----#

mc_comp('/shared/utils/info_string', %ARGS, STORE=>\$info_string);
my $giftcard = "10.20.1.53";
my $url = "http://$giftcard/process/charge?$info_string";

#----- Get the LOC based on the u_id. Need these to redirect back to m
all -----#

my $LOC = mc_comp( 'get_LOC', u_id=>$ARGS{'u_id'} );

#----- Set various URLs -----#

my $return = mc_comp('/shared/http/lwp_call', url=>$url, timeout=>90, retry=>1
, data=>\%data);
my $receipt = "https://secure.luciditygifts.com/giftcard/add_receipt?$info_stri
ng&LOC=$LOC";
my $small_url = "http://www.luciditygifts.com/giftcard/mall_shops?LOC=$LOC";

$receipt =~ s/\s//g;
$small_url =~ s/\s//g;
</%init>

if ($return != 200 || $data{success} == 0) {
    <& error_template, %ARGS, %data, top=>1&>
    <& error, %ARGS, %data, return=>$return, url=>$url, error_id=>$info_string
    &>
    <& error_template, bottom=>1&>
    return 1;
} elsif ($data{success} == 3) {
    my (%missing);
    $missing{status} = 1;
    $missing{card_number} = "$tag_start Your Card was declined, please try anot
her card. $tag_end";
    <& add_template, %ARGS, %data, top=>1&>
    <& add_info, missing=>\%missing, %ARGS &>
    <& add_template, %ARGS, %data, bottom=>1&>
    return 1;
} elsif ($data{success} == 2) {
    % $ARGS{queue} = 1;
} elsif ($data{success} == 4) {
    % $ARGS{duplicate} = 1;
}

<SCRIPT LANGUAGE="JavaScript">
<!--
    // Reload top frame with receipt
    parent.frames[0].location.href = "<% $receipt %>";
/-->
</SCRIPT>
```

```
% if( $returnurl ) {  
    <SCRIPT LANGUAGE="JavaScript">  
        <!--  
            //Reload this frame with return URL  
            parent.frames[1].location.href = "<$$returnurl%>";  
        -->  
    </SCRIPT>  
  
% } else {  
    <SCRIPT LANGUAGE="JavaScript">  
        <!--  
            //Reload this frame with mall URL  
            parent.frames[1].location.href = "<$$mall_url%>";  
        -->  
    </SCRIPT>  
  
% }  
  
<%ARGS>  
$returnurl=>undef  
$tag_start=>'<FONT COLOR=red>  
$tag_end=>'</FONT>'  
</%ARGS>
```



```

<%ARGS>
$top=>undef
$bottom=>undef
$title=>'Credit Card information'
$amount=>undef
$damt=>undef
$store=>undef
$confirm=>undef
$balance=>undef
$process=>"INFO"
</%ARGS>
<%INIT>
my $image_base = "https://fund.luciditygifts.com/images/giftcard";
my $new_balance = sprintf "%-2.2f", $balance+$amount;

</%INIT>
% if ($top) {

<html>
<body bgcolor=#FFFFFF background=<% $image_base %>/ribbonsbgc2.gif topmargin=4>

<center>
<table width=700 bgcolor=ffffff border=0 cellpadding=2 cellspacing=0>
<tr>
<td valign=bottom>
<img src=<% $image_base %>/addheadernew.gif><!--<% $image_base %>/gold_logobt.gif>
</td>
<td align=center valign=bottom><img src=<% $image_base %>/bluevertl.gif width=7
height=70>
</td>
<td valign=bottom align=left><font color=000000 size=5 face=arial,Helvetica><font
size=4 color=003399 face=arial,Helvetica>
% if ($process eq "CONFIRM") {
    You will be <b>adding $<% $amount %></b> Bluemountain Gold Gift Certifica
    te
    to yield a total balance of $<% $new_balance %>.</font>
% } else {
%     if ($store) {
        <br><b>You are $<% $amount %> short for your purchase at <% $store %>.</b>
<br>
        To add money to your Blue Gold, fill out the payment form below.
        You will then be able to shop again.
        You will then be returned to <% $store %> to complete your check-out!
    } else {
        To add money to your Blue Gold, fill out the payment form below.
        You will then be able to shop again.
    }
% }

</td>
</tr>
<tr>
<td colspan=3><img src=<% $image_base %>/bluehorizh.gif border=0></td></tr>
</table>

<table width=700 bgcolor=#FFFFFF border=0 cellspacing=0 cellpadding=0>
<tr><td align=center>

```

```
</font></font>
</td></tr>
</table>
```

```
<table width=700 bgcolor=#66ccff border=0 cellspacing=0 cellpadding=0>
<tr><td colspan=4 width=100%>&nbsp;</td>
</tr>
</table>
```

```
<TABLE BORDER=0 BGCOLOR="#66CCFF" CELLSPACING=0 CELLPADDING=4 WIDTH=700><TR><TD>
```

```
<FORM ACTION="add" METHOD=post>
% }
```

```
% if ($bottom) {
```

```
</CENTER>
<!BIG></TD></TR></TABLE>
&nbsp;<br>
```

```
</BODY>
</HTML>
```

```
%if (0) {
```

```
New end
```

```
<TABLE WIDTH="700" BGCOLOR="#99cccc" BORDER="0" CELLPADDING="0" CELLSPACING="0">
<TR>
```

```
<TD VALIGN="bottom" ALIGN="left" WIDTH="40" BGCOLOR="#ffffff">
```

```
<IMG SRC="<%$image_base%/giftcard/gold_lbcorner.gif" BORDER="0" WIDTH="40"
HEIGHT="53" ALT="" HSPACE="0" VSPACE="0">
```

```
</TD>
```

```
<TD WIDTH=620>&nbsp;</TD>
```

```
<TD WIDTH=40 VALIGN="bottom" ALIGN="right" BGCOLOR="#ffffff">
```

```
<IMG SRC="<%$image_base%/giftcard/gold_rbcorn.gif" border="0" WIDTH="40"
HEIGHT="53" ALT="" HSPACE="0" VSPACE="0">
```

```
</TD>
```

```
</TR>
</TABLE>
```

```
</BODY>
```

```
</HTML>
```

```
}
```

```
}
```

```
<%INIT>
my (%missing) = (MissingCount=>0);
my (%validation) = (
    to_email      => 'return ($ARGS{$key} =~ /\w+@[ \w|\.|+\/]'/,
    card_exp_month => 'return ($ARGS{$key} =~ /\d{2}/)',
    card_exp_year  => 'return ($ARGS{$key} =~ /\d{4}/',
    amount         => 'return ($ARGS{$key} =~ /^( \d+\.? \d* | \. \d+)$/ ',
    amount         => 'return ($ARGS{$key} < 500',
);

my (%check_length) = (
    to_email      => 100,
    card_first_name => 25,
    card_last_name  => 25,
    company         => 50,
    card_street     => 60,
    card_apt        => 5,
    card_city       => 50,
    card_state      => 50,
    card_zipcode    => 10,
    card_country    => 50,
    area_code       => 3,
    phone           => 10,
    ext             => 6,
);

my (@validation) = qw(
    amount
    card_first_name card_last_name to_email card_type card_street
    card_city card_state card_zipcode card_country
);

foreach my $key (@validation) {
    unless ($ARGS{$key}) {
        $missing{$key} = $star;
        $missing{MissingCount}++;
    }
}

foreach my $key (keys %validation) {
    unless (eval $validation{$key}) {
        $missing{$key} = $star;
        $missing{MissingCount}++;
    }
}

foreach my $key (keys %check_length) {
    if (length ($ARGS{$key}) > $check_length{$key}) {
        $missing{$key} = "$tag_start $key can only be $check_length{$key} characters $tag_end";
        $missing{MissingCount}++;
    }
}

if ($ARGS{card_number}) {
    unless (mc_comp('/shared/credit/ccValidate',
        CCNumber=>$ARGS{card_number}, CCCardType=>$ARGS{card_type})) {
        $missing{card_number} = "$tag_start This does not appear to be a correct Card Number. $tag_end";
    }
}
```

```
        $missing{MissingCount}++;
    }
} else {
    $missing{card_number} = $star;
    $missing{MissingCount}++;
}

if ($ARGS{amount} > $max_amount) {
    $missing{amount} = "$tag_start Maximum amount is \$$max_amount. $tag_end";
    $missing{MissingCount}++;
} elsif ($ARGS{amount} < $min_amount) {
    $missing{amount} = "$tag_start Minimum amount is \$$min_amount. $tag_end";
    $missing{MissingCount}++;
}

$missing{status} = $missing{MissingCount} ? 1 : 0;
return %missing;

</%INIT>
<%ARGS>
$tag_start=>'<FONT COLOR=red>'
$tag_end=>'</FONT>'
$star=>"<FONT COLOR=red>*</FONT>"
$max_amount=>500
$min_amount=>'.01'
/<%ARGS>
```

```
<html><header>
<META http-equiv=refresh content="90; url=<%=url%>">
</header>
<body bgcolor=#ffffff topmargin=0>
<center>
<table width=680 bgcolor=FFFFFF>
<tr>
<td align=left><img src=/images/giftcard/headerw.gif></td>
<td width=65% align=center><font size=5> <b>Please wait</b> while the new funds
are added to your Blue Mountain Gold Certificate.
</font></b></td>
</tr>
<tr>
<td colspan=2 align=center><img src=/images/giftcard/bluehorizh.gif></td>
</tr>
<tr>
<td colspan=2 align=center> <!-- bgcolor=99cccc -->
<font size=5 color=330099>
<b>This process will take up to two minutes.</b><br>
Once your Blue Gold balance has been successfully updated, you will be automatic
ally
returned where you left off.<br></td></tr><tr>
<td colspan=2 align=center><img src=/images/giftcard/bluehorizh.gif></td></tr><t
r><td colspan=2>
<nbsp><p><img src=/images/giftcard/hhogcp.gif><br>
</font>
</td>
</table>
</body></html>

%once>
use URI::Escape;
</%once>
%init>
$returnurl = uri_escape($returnurl, "^A-Za-z0-9");
my $url= "add_receipt?LOC=$LOC&u_id=$u_id&amount=$amount&balance=$balance&order_
_id=$order_id&returnurl=$returnurl";
</%init>
%args>
$returnurl=>' '
$LOC=>' '
$u_id=>' '
$amount=>' '
$balance=>' '
$order_id=>' '
</%args>
```

```
<html><header><title>Blue Gold Bar</title></header>
<body background="bluevert1.gif" TOPMARGIN=4 LEFTMARGIN=3 MARGINHEIGHT=3 MARGINW
IDTH=3>

<table cellpadding=0 cellspacing=0 width=99% height=60>
<tr><td width=145 valign=top rowspan=2>


</td>
<td valign=middle align=center rowspan=2>
  <table>
    <tr><td><img src=balabs.gif>
    </td><td valign=top align=left><font size=5 color=#003399 face=arial,
helvetica> <!color=003366> <!color=#006699 size=5><b>$40.49</b></font></td></tr>
  </table>
</td>

<td valign=middle align=center><font color=003399 size=2 face=arial, helvetica>
b><i>
      To change stores,<br> click on "Shop Menu."
    </i></b></font></td>

<td align=right>
  <table cellpadding=0 cellspacing=0 border=0>
    <tr>
      <td rowspan=2 valign=middle><a href=mallblue.html target=BOTTOM_F
FRAME><img src=shopblue.gif border=0><!flatshop.gif border=0 alt=Shops><!bigyell>
/a></td>
      <td rowspan=2 width=10>&nbsp;</td>
      <td align=left valign=middle><a href=newaddblue.html target=BOTTO
M_FRAME><img src=addbuttt.gif border=0 alt=Add></a></td><td width=5>&nbsp;</td><
td>
      <img src=coinsplash.gif width=38 height=32 hspace=0 vspace=0></td>
    </tr><tr>
      <td colspan=3 align=left><a href=infohelp.html target=BOTTOM_FRAM
E><img src=infohelpbutt.gif border=0 alt=Help></a></td></tr>
  </table>

</td>

</tr>
</table>

</body></html>
```

```
<HTML><HEAD>
<TITLE>Blue Mountain Arts Charity Donation Page</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" background="sky_bg.gif">
<center>
<table border="0" cellpadding="0" cellspacing="0" width="700">
<tr>
    <td><IMG SRC="attcorner.gif" WIDTH="80" HEIGHT="73"></td>
    <td bgcolor="#cc99cc"><IMG SRC="bmacharitytitlesm.gif" WIDTH="252" HEIGH
T="59"></td>
    <td bgcolor="#cc99cc" align="left" valign="middle"></td>
    <td><IMG SRC="rtcorner.gif" WIDTH="40" HEIGHT="73"></td>
</tr>
</table>
<table border="0" cellpadding="0" cellspacing="0" width="700">
<tr>
    <td width="15" bgcolor="#cc99cc" height="50">&nbsp;</td>
    <td width="670" bgcolor="#cccccc">&nbsp;</td>
    <td width="15" bgcolor="#cc99cc">&nbsp;</td>
</tr>
</table>
<table border="0" cellpadding="0" cellspacing="0" width="700">
<tr>
    <td width="30">
</td>
    <td width="640" bgcolor="#cc99cc">&nbsp;</td>
    <td width="30">
</td>
</tr>
</table>
</center>
</BODY>
</HTML>
```

```
<%PERL_INIT>

my @types = (
    { 'code'=>'Amex', 'name'=>'American Express' },
    { 'code'=>'Visa', 'name'=>'Visa' },
);

return( @types );

</%PERL_INIT>
```


<%doc>

Passing a Credit Card number and a Credit Card type
Check the validity of a card.

Return 0 for bad card,

1 for a good card.

System cleans the number, does various logical checks
and checks the cards actual validity algorithym.

I've only tested on one visa number.

</%doc>

<%init>

my \$result;

my \$offset;

my \$checkBit = substr(\$CCNumber, 0, 2);

#remove noise

\$CCNumber =~ s/[s\-\-]/g;

return 0 if \$CCNumber !~ /\d+\$/;

my \$CCLength = length \$CCNumber;

if (lc(\$CCCardType) eq "visa") {

return 0 unless \$CCLength == 13 or \$CCLength == 16;

return 0 unless \$checkBit =~ /^4\d\$/;

\$offset = 1;

} elsif (lc(\$CCCardType) eq "amex") {

return 0 unless \$CCLength == 15;

return 0 unless \$checkBit == 34 or \$checkBit == 37;

\$offset = 0;

elsif (lc(\$CCCardType) eq "mastercard") {

return 0 unless \$CCLength == 16;

return 0 unless \$checkBit =~ /^5\d\$/;

\$offset = 1;

elsif (lc(\$CCCardType) eq "discovery") {

return 0 unless \$CCLength == 16;

return 0 unless \$checkBit == 60;

\$offset = 1;

else {

return 0;

}

foreach my \$i (0 .. (\$CCLength-1)) {

my \$bit = substr(\$CCNumber, \$i, 1);

#determine if we should double the bit

my \$double = (\$i+\$offset) % 2;

my \$bitSum = \$bit + (\$double * \$bit);

\$bitSum = substr(\$bitSum,0,1) + substr(\$bitSum,1,1) if \$bitSum >= 10;

\$result += \$bitSum;

return !(\$result % 10);

</%init>

<%args>

\$CCNumber

\$CCCardType

</%args>

```
<%doc>
Executes direct submission of a payment to CyberCash CashRegister

Args:
%sender
- what are the keys of this hash?
%cc
- what are the keys of this hash?
%order
- what are the keys of this hash?

Returns:
\%POP
- what are the keys of this hash?

\%tokenList
- what are the keys of this hash?

Example:
mc_comp('/shared/credit/charge', sender=>%sender, cc=>%sender, order=>%order)

</%doc>
<%args>
%sender
%cc
%order
/%args>
<%init>
use CyberC::CCMckLib3_2;
use CyberC::CCMckDirectLib3_2;
use CyberC::CCMckErrno3_2;

my (%POP,%tokenList );

# Initialize the Global Configuration array in CCMckLib3_2
# JW - change setting of $ConfigFile variable based on where it will eventually
# reside
my $ConfigFile='/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/conf/merch
ant_conf';
my $status = CyberC::CCMckLib3_2::InitConfig($ConfigFile);

if ($status != 0) {
    $POP{status} = MCKGetErrorMessage($status);
    return (\%POP, \%tokenList);
}

CyberC::CCMckLib3_2::CCDebug("Entering $0\n");

my $cybercashId = $CyberC::CCMckLib3_2::Config("CYBERCASH_ID");
my $hashSecret = $CyberC::CCMckLib3_2::Config("HASH_SECRET");
my $ccpsHost = $CyberC::CCMckLib3_2::Config("CCPS_HOST");
my $templateDir = $CyberC::CCMckLib3_2::Config("TEMPLATE_DIR");
my $redirectURL = $CyberC::CCMckLib3_2::Config("REDIRECT_URL");

# A Note about message fields in paymentNVList.
#   paymentNVList holds message parameters
#   that will be passed to the Cash Register
#
# These parameters are broken into three blocks according to
# a prefix in the argument field name.
```

```

#
# prefix          use
#
# mo.  -- data driving the Cash Register
#         this include Payment data and some operational options
# cpi.  -- info about the Customer's "Payment Instrument: (credit/check
)
# mf.  -- anything additional you want to pass on
#         to the Fulfillment Center

my %paymentNVList;
$paymentNVList{'mo.cybercash-id'} = $cybercashId;
$paymentNVList{'mo.version'} = $CyberC::CCMckLib3_2::MCKversion; #CCPS message
version

# The MO and CPI blocks are not signed, since
# the message will be encrypted

$paymentNVList{'mo.signed-cpi'} = "no";

# Load payment instrument info from form

if( !$order{order_id} ){
    $paymentNVList{'mo.order-id'} = mc_comp('/shared/utills/unique_id');
} else {
    $paymentNVList{'mo.order-id'} = $order{order_id};

$paymentNVList{'mo.price'}           = 'USD '.$order{'amount'};
$paymentNVList{'cpi.card-number'}    = $cc{'card_number'};
$paymentNVList{'cpi.card-exp'}       = $cc{'card_exp_month'}.'/'.'$cc{'card_ex
p_year'};
$paymentNVList{'cpi.card-name'}      = $sender{'first_name'}.' '.$sender{'las
t_name'};
$paymentNVList{'cpi.card-address'}   = $sender{'street'}.' '.$sender{'apt'};
$paymentNVList{'cpi.card-city'}      = $sender{'city'};
$paymentNVList{'cpi.card-state'}     = $sender{'state'};
$paymentNVList{'cpi.card-zip'}       = $sender{'zip'};
$paymentNVList{'cpi.card-country'}   = $sender{'country'};

# This is a CREDIT CARD payment
# DON'T TOUCH THIS! This is the CR script that will process your payment
my $script = "directcardpayment.cgi";

#####
# No signatures required to talk to the Cash Register
# NO SSL required to talk to the Cash Register either...
# Since the whole thing is encrypted, crooks can't mess with it
# and can't read it either!
#####

# merchant-id gets added to the URL by the connection routine.
# Don't add it here...

my $paymentURL = "$ccpsHost$script";

# At this point, all of the data that needs to go into the Payment message
# should have been collected: So, we can call the Cash Register
my ($POPref, $tokenListref) = CyberC::CCMckDirectLib3_2::doDirectPayment( $payme
ntURL, \%paymentNVList);

```

```
%POP = %$POPref;
%tokenList = %$tokenListref;
my $POPstr = $tokenList{'POP'};

if ($POP{'pop.status'} =~ /^success-duplicate/ ) {
    #add a message to the template
    $tokenList{'DUPLICATE_MSG'}
        = "<p>Please note that this order is a duplicate of one complete
d previously</p>\n";
}
else {
    $tokenList{'DUPLICATE_MSG'} = " "; # put in a blank to ensure substituti
on
}

CyberC::CCMckLib3_2::CCDebug("Exiting $0\n");

return (\%POP, \%tokenList);

</%init>
```

<%PERL_DOC>

Component to encapsulate a call to app server to charge credit card.

Should return:

- whether charge was accepted/declined
- information about the order
- any error information generated

Which of the arguments are necessary?

</%PERL_DOC>

<%PERL_ARGS>

\$queue=>0

\$order_id=>''

\$amount=>'2001.00'

\$rcv_email=>'jwintert@optonline.net'

\$rcv_first_name=>'Jennifer'

\$rcv_last_name=>'Winterton'

\$snd_first_name=>'Jason'

\$snd_last_name=>'Taylor'

\$snd_email=>'jsmith@jsmith.com'

\$card_type=>'visa'

\$card_number=>'4111111111111111'

\$card_exp_month=>'12'

\$card_exp_year=>'04'

\$street=>'126-128 Church Street'

\$apt=>''

\$city=>'San Francisco'

\$state=>'CA'

\$zip_code=>'94114-1111'

\$country=>'USA'

\$area_code=>''

\$phone=>''

</%PERL_ARGS>

<%PERL_INIT>

----- Logically split data -----#

my \$receiver_data = {

- 'email' => \$rcv_email,
- 'first_name' => \$rcv_first_name,
- 'last_name' => \$rcv_last_name,

};

my \$sender_data = {

- 'first_name' => \$snd_first_name,
- 'last_name' => \$snd_last_name,
- 'email' => \$snd_email,

};

my \$cc_data = {

- 'card_type' => \$card_type,
- 'card_number' => \$card_number,
- 'card_exp_month' => \$card_exp_month,
- 'card_exp_year' => \$card_exp_year,
- 'street' => \$street,
- 'apt' => \$apt,
- 'city' => \$city,
- 'state' => \$state,
- 'zip_code' => \$zip_code,
- 'country' => \$country,
- 'area_code' => \$area_code,

```
};
    'phone'          => $phone,
};

my $order_data = {
    'order_id'       => $order_id,
    'amount'         => $amount,
};

#----- Check for mandatory fields -----#
# Insert error checking code here
#----- Check for correct formatting of data -----#

if( $cc_data->{'card_exp_year'} =~ /\d{3,4}$/ ) {
    $cc_data->{'card_exp_year'} =~ s/(.*) (\d\d)$/$2/;
}

#----- Write information to database -----#
# mc_comp( '/api/record_data/writeSenderData',    data=>$sender_data );
# mc_comp( '/api/record_data/writeRecieverData',  data=>$receiver_data );
# mc_comp( '/api/record_data/writeCCData',        data=>$cc_data );

#----- Call to CyberCash module to charge -----#

my ( $POPref,$tokenListref ) = mc_comp( '/lucidity/app/secure/credit_auth', send
er_data=>$sender_data,
                                     cc_data=>$cc_data, order_data=>$order_data );

my %POP = %$POPref;
my %tokenList = %$tokenListref;

#----- Place returned information in data structure -----#
my %result = {
    'status' => $POP{'pop.status'},
    'error' => {
        'error_code' => '',
        'error_message' => '',
    },
    'order' => {
        'order_no' => '',
        'order_amount' => '',
    },
};

#----- Based on return codes, take some actions -----#

if( $result{'status'} =~ /failure-hard|failure-q-or-cancel|failure-q-or-discard|
failure-swversion/ ) {
    # queue the transaction
    $result{'status'} = 'queued';
}

#----- Record the transaction data -----#

# Get the transaction data out of the return values here
```

```
# mc_comp( '/api/record_data/writeTransactionData', data=>%transaction_data );

#return( \%result );
return ( \%POP );

</%PERL_INIT>
```

```
<%perl_doc>
this component is used for a direct submission of a payment to CyberCash CashReg
ister
</%perl_doc>
<%args>
$order_id=>'
$amount=>0
$first_name
$last_name
$card_number
$card_exp_month
$card_exp_year
$street=>'
$apt=>'
$city=>'
$state=>'
$zip_code
$country=>'
</%args>
<%once>
use CyberC::CCMckLib3_2;
use CyberC::CCMckDirectLib3_2;
use CyberC::CCMckErrno3_2;

my ($sender, %cc, %order, %POP, %tokenList );
# Initialize the Global Configuration array in CCMckLib3_2
# JW - change setting of $ConfigFile variable based on where it will eventually
# reside and confirm giftcat names
my $ConfigFile;
if ($ARGS{giftcat} eq 'charities'){
    $ConfigFile = '/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/con
f/merchant_conf';
}
elsif ($ARGS{giftcat} eq 'giftcert'){
    $ConfigFile = '/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/con
f/shine_conf';
}
my $status = CyberC::CCMckLib3_2::InitConfig($ConfigFile);
</%once>
<%init>
if ($status != 0) {
    $POP{status} = MCKGetErrorMessage($status);
    return (\%POP, \%tokenList);
}

CyberC::CCMckLib3_2::CCDebug("Entering $0\n");

my $cybercashId = $CyberC::CCMckLib3_2::Config{"CYBERCASH_ID"};
my $hashSecret = $CyberC::CCMckLib3_2::Config{"HASH_SECRET"};
my $ccpsHost = $CyberC::CCMckLib3_2::Config{"CCPS_HOST"};
my $templateDir = $CyberC::CCMckLib3_2::Config{"TEMPLATE_DIR"};
my $redirectURL = $CyberC::CCMckLib3_2::Config{"REDIRECT_URL"};

# A Note about message fields in paymentNVList.
#   paymentNVList holds message parameters
#   that will be passed to the Cash Register
#
# These parameters are broken into three blocks according to
# a prefix in the argument field name.
```



```

#
# prefix          use
#
# mo.  -- data driving the Cash Register
#          this include Payment data and some operational options
# cpi.  -- info about the Customer's "Payment Instrument: (credit/check
)
# mf.  -- anything additional you want to pass on
#          to the Fulfillment Center

my %paymentNVList;
$paymentNVList{'mo.cybercash-id'} = $cybercashId;
$paymentNVList{'mo.version'} = $CyberC::CCMckLib3_2::MCKversion; #CCPS message
version

# The MO and CPI blocks are not signed, since
# the message will be encrypted

$paymentNVList{'mo.signed-cpi'} = "no";

# load payment instrument info from form

if( !defined($order_id) || !$order_id ){
    $paymentNVList{'mo.order-id'} = mc_comp('/shared/utills/unique_id');
} else {
    $paymentNVList{'mo.order-id'} = $order_id;

$paymentNVList{'mo.price'}           = 'USD '.$amount;
$paymentNVList{'cpi.card-number'}    = $card_number;
$paymentNVList{'cpi.card-exp'}       = $card_exp_month.'/'.$card_exp_year;
$paymentNVList{'cpi.card-name'}      = $first_name.' '.$last_name;
$paymentNVList{'cpi.card-address'}   = $apt ? $street.' '.$apt : $street;
$paymentNVList{'cpi.card-city'}      = $city;
$paymentNVList{'cpi.card-state'}     = $state;
$paymentNVList{'cpi.card-zip'}       = $zip_code;
$paymentNVList{'cpi.card-country'}   = $country;

# This is a CREDIT CARD payment
# DON'T TOUCH THIS! This is the CR script that will process your payment
my $script = "directcardpayment.cgi";

my $paymentURL = "$ccpsHost$script";

#At this point, all of the data that needs to go into the Payment message
# should have been collected: So, we can call the Cash Register

my ($POPref, $tokenListref) = CyberC::CCMckDirectLib3_2::doDirectPayment( $payme
ntURL, \%paymentNVList);

CyberC::CCMckLib3_2::CCDebug("Exiting $0\n");

return ($POPref);
</%init>

```

```
<%init>
my $cookie;
if ($ARGS{name} & $ARGS{value}) {

    # Set cookie
    $cookie = new CGI::Cookie
    (
        -name    => $ARGS{name},
        -value   => $ARGS{value},
        -expires => $ARGS{expires},
        -domain  => $ARGS{domain},
        -path    => $ARGS{path},
    );

    $r->header_out('Set-Cookie' => $cookie);
}
else {

    # Get cookie
    return fetch CGI::Cookie ($r->header_in('Cookie'));
}
</%init>
```

```
<%perl_doc>
this component is used for a direct submission of a payment to CyberCash CashReg
ister
<%/perl_doc>
<%args>
<%PERL_ARGS>
$sender_data=>' '
$cc_data=>' '
$order_data=>' '
</%PERL_ARGS>
<%init>
use CyberC::CCMckLib3_2 qw($MCKversion %Config URLdecodeForm
    InitConfig CCErrors CCDebug CCDebug2 GetQuery
    PrintTemplate LoadBlockFromQuery
    BuildSignature );

use CyberC::CCMckDirectLib3_2 qw(doDirectPayment);

use CyberC::CCMckErrno3_2 qw(MCKGetErrorMessage $E_NoErr
    $E_Fail_Notif
    $E_Payment_Failed
    $E_No_Receipt
    $E_MO_Signature
    $E_MF_Signature
    $E_No_Template
    );

my (%POP,%tokenList);

# Initialize the Global Configuration array in CCMckLib3_2
# JW - change setting of $ConfigFile variable based on where it will eventually
# reside
my $ConfigFile='/www/servers/lucidity/app/test-mck/mck-cgi/conf/merchant_conf'
my $status = &InitConfig($ConfigFile);

if ($status != 0) {
    $POP{status} = &MCKGetErrorMessage($status);
    return (\%POP, \%tokenList);
}

&CCDebug("Entering $0\n");

my $cybercashId = $Config{"CYBERCASH_ID"};
my $hashSecret = $Config{"HASH_SECRET"};
my $ccpsHost = $Config{"CCPS_HOST"};
my $templateDir = $Config{"TEMPLATE_DIR"};
my $redirectURL = $Config{"REDIRECT_URL"};

# A Note about message fields in paymentNVList.
#   paymentNVList holds message parameters
#   that will be passed to the Cash Register
#
# These parameters are broken into three blocks according to
# a prefix in the argument field name.
#
# prefix          use
#
# mo.    -- data driving the Cash Register
#          this include Payment data and some operational options
#
# cpi.    -- info about the Customer's "Payment Instrument: (credit/check
```

```
)
# mf. -- anything additional you want to pass on
# to the Fulfillment Center

my %paymentNVList;
$paymentNVList{'mo.cybercash-id'} = $cybercashId;
$paymentNVList{'mo.version'} = $MCKversion; #CCPS message version

# The MO and CPI blocks are not signed, since
# the message will be encrypted

$paymentNVList{'mo.signed-cpi'} = "no";

# load payment instrument info from form

if( !defined($order_data->{order_id}) || !$order_data->{order_id} ){
    $paymentNVList{'mo.order-id'} = mc_comp('/shared/credit_card/uniqueId');
} else {
    $paymentNVList{'mo.order-id'} = $order_data->{order_id};
}

$paymentNVList{'mo.price'} = 'USD '.$order_data->{'amount'};
$paymentNVList{'cpi.card-number'} = $cc_data->{'card_number'};
$paymentNVList{'cpi.card-exp'} = $cc_data->{'card_exp_month'}.'/'.$cc_data->{'card_exp_year'};
$paymentNVList{'cpi.card-name'} = $sender_data->{'first_name'}.' '.$sender_data->{'last_name'};
$paymentNVList{'cpi.card-address'} = $sender_data{'street'}.' '.$sender_data{'apt'};
$paymentNVList{'cpi.card-city'} = $sender_data{'city'};
$paymentNVList{'cpi.card-state'} = $sender_data{'state'};
$paymentNVList{'cpi.card-zip'} = $sender_data{'zip'};
$paymentNVList{'cpi.card-country'} = $sender_data{'country'};

# This is a CREDIT CARD payment
# DON'T TOUCH THIS! This is the CR script that will process your payment
my $script = "directcardpayment.cgi";

#####
# No signatures required to talk to the Cash Register
# NO SSL required to talk to the Cash Register either...
# Since the whole thing is encrypted, crooks can't mess with it
# and can't read it either!
#####

# merchant-id gets added to the URL by the connection routine.
# Don't add it here...

my $paymentURL = "$ccpsHost$script";

#At this point, all of the data that needs to go into the Payment message
# should have been collected: So, we can call the Cash Register

my ($POPref, $tokenListref) = &doDirectPayment( $paymentURL, \%paymentNVList);

$POP = %$POPref;
$tokenList = %$tokenListref;
my $POPstr = $tokenList{'POP'};
```

```
if ($POP{'pop.status'} =~ /^success-duplicate/ ) {  
    #add a message to the template  
    $tokenList{'DUPLICATE_MSG'}  
        = "<p>Please note that this order is a duplicate of one complete  
d previously</p>\n";  
}  
else {  
    $tokenList{'DUPLICATE_MSG'} = " "; # put in a blank to ensure substituti  
on  
}  
  
&CCDebug("Exiting $0\n");  
  
return (\%POP, \%tokenList);  
  
<%/init>
```

```
<%perl_doc>
this component is used for a direct submission of a payment to CyberCash CashReg
ister
</%perl_doc>
<%args>
$sender_data=>' '
$cc_data=>' '
$order_data=>' '
</%args>
<%init>
use CyberC::CCMckLib3_2;
use CyberC::CCMckDirectLib3_2;
use CyberC::CCMckErrno3_2;

my (%sender, %cc, %order, %POP,%tokenList );
$sender = $$sender_data;
%cc = %$cc_data;
$order = %$order_data;

# Initialize the Global Configuration array in CCMckLib3_2
# JW - change setting of $ConfigFile variable based on where it will eventually
reside
my $ConfigFile='/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/conf/merch
ant_conf';
my $status = CyberC::CCMckLib3_2::InitConfig($ConfigFile);

if ($status != 0) {
    $POP{status} = MCKGetErrorMessage($status);
    return (\%POP, \%tokenList);
}

CyberC::CCMckLib3_2::CDebug("Entering $0\n");

my $cybercashId = $CyberC::CCMckLib3_2::Config{"CYBERCASH_ID"};
my $hashSecret = $CyberC::CCMckLib3_2::Config{"HASH_SECRET"};
my $ccpsHost = $CyberC::CCMckLib3_2::Config{"CCPS_HOST"};
my $templateDir = $CyberC::CCMckLib3_2::Config{"TEMPLATE_DIR"};
my $redirectURL = $CyberC::CCMckLib3_2::Config{"REDIRECT_URL"};

# A Note about message fields in paymentNVList.
#   paymentNVList holds message parameters
#   that will be passed to the Cash Register
#
# These parameters are broken into three blocks according to
# a prefix in the argument field name.
#
# prefix          use
#
# mo.    -- data driving the Cash Register
#          this include Payment data and some operational options
# cpi.    -- info about the Customer's "Payment Instrument: (credit/check
)
# mf.    -- anything additional you want to pass on
#          to the Fulfillment Center

my %paymentNVList;
$paymentNVList{'mo.cybercash-id'} = $cybercashId;
$paymentNVList{'mo.version'} = $CyberC::CCMckLib3_2::MCKversion; #CCPS message
version
```

```
# The MO and CPI blocks are not signed, since
# the message will be encrypted
```

```
$paymentNVList{'mo.signed-cpi'} = "no";
```

```
# load payment instrument info from form
```

```
if( !defined($order{order_id}) || !$order{order_id} ){
    $paymentNVList{'mo.order-id'} = mc_comp('/shared/utills/uniqueId');
} else {
    $paymentNVList{'mo.order-id'} = $order{order_id};
}
```

```
$paymentNVList{'mo.price'}           = 'USD '.$order{'amount'};
$paymentNVList{'cpi.card-number'}    = $cc{'card_number'};
$paymentNVList{'cpi.card-exp'}       = $cc{'card_exp_month'}.'/'.$cc{'card_ex
p_year'};
$paymentNVList{'cpi.card-name'}      = $sender{'first_name'}.' '.$sender{'las
t_name'};
$paymentNVList{'cpi.card-address'}   = $sender{'street'}.' '.$sender{'apt'};
$paymentNVList{'cpi.card-city'}      = $sender{'city'};
$paymentNVList{'cpi.card-state'}     = $sender{'state'};
$paymentNVList{'cpi.card-zip'}       = $sender{'zip'};
$paymentNVList{'cpi.card-country'}   = $sender{'country'};
```

```
# This is a CREDIT CARD payment
# DON'T TOUCH THIS! This is the CR script that will process your payment
my $script = "directcardpayment.cgi";
```

```
#####
# No signatures required to talk to the Cash Register
# NO SSL required to talk to the Cash Register either...
# Since the whole thing is encrypted, crooks can't mess with it
# and can't read it either!
#####
```

```
# merchant-id gets added to the URL by the connection routine.
# Don't add it here...
```

```
my $paymentURL = "$ccpsHost$script";
```

```
#At this point, all of the data that needs to go into the Payment message
# should have been collected: So, we can call the Cash Register
my ($POPPref, $tokenListref) = CyberC::CCMckDirectLib3_2::doDirectPayment( $payme
ntURL, \%paymentNVList);
$POP = %$POPPref;
$tokenList = %$tokenListref;
my $POPstr = $tokenList{'POP'};
```

```
if ($POP{'pop.status'} =~ /^success-duplicate/ ) {
    #add a message to the template
    $tokenList{'DUPLICATE_MSG'}
        = "<p>Please note that this order is a duplicate of one complete
d previously</p>\n";
}
else {
    $tokenList{'DUPLICATE_MSG'} = " "; # put in a blank to ensure substituti
on
}
```

```
CyberC::CCMckLib3_2::CCDebug("Exiting $0\n");  
return (\\%POP, \\%tokenList);  
  
</%init>
```


[illegible]

We are sorry, there appears to have been an error with your order.

Please bookmark <A HREF="<% \$resolve_url %>">this link and check back later.

Thanks,

The Management

<%init>
my \$resolve_url = "https://secure.luciditygifts.com/charity/resolve?\$error_id";
mc_comp('/shared/utills/log_error', url=>\$url, return=>\$return, giftcat=>\$ARGS{gi
ftcat}, %ARGS);
</%init>
<%args>
\$url=>undef
\$return=>undef
\$error_id=>undef
</%args>

```
%if ($stop) {
<HTML>
<HEAD>
<TITLE>Add Funds to Your Bluemountain Gold</TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#990000" VLINK="#FF9900">

<CENTER>

<TABLE BORDER=0 BGCOLOR="#ffffff" CELLSPACING=0 CELLPADDING=4 WIDTH=700><!FFFFCC
>
  <TR>
    <TD><IMG SRC="/images/giftcard/gold_logob.gif" ALT="" WIDTH=164 HEIGHT=109 H
SPACE=0 VSPACE=0></TD>

    <TD VALIGN=middle ALIGN=left>
      <FONT FACE="Helvetica, Arial"Q SIZE=2 COLOR="#003399">
        Unfortunately, something bad seems to have happened. Please try again in a
        few minutes. We
        have been notified and will look into it immediately.

      <br>
      </FONT>
    <FONT FACE="Helvetica, Arial"Q SIZE=2 COLOR="#003399"> <center>
      <i>Bluemountain respects your privacy, and provides you with secure transactions
      </i>
    </center>
    </TD>
  </TR>
</TABLE><center>

<BIG><TABLE BORDER=0 BGCOLOR="#99cccc" CELLSPACING=0 CELLPADDING=4 WIDTH=700><T
B><TD>

<FORM ACTION="charge" METHOD=post>
%}

%if ($bottom) {
<BIG></TD></TR></TABLE>
</BODY>
</HTML>
%}
<%args>
  <stop>=>undef
  <bottom>=>undef
</%args>
```

-----BEGIN CERTIFICATE-----

MIIC8zCCAlYgAwIBAgIDAKtCMA0GCSqGSIb3DQEBAUAMIHEMQswCQQYDVQQGEwJa
QTEVMBMGA1UECBMMV2VzdGVybiBDYXB1MRlWEAYDVQQHEw1DYXB1IFRvd24xHTAb
BgNVBAoTFFRoYXN0ZSBDb25zdWx0aW5nIGNjMSgwJG9YDVQQLEx9DZXJ0aWZpY2F0
aW9uIFNlcnZpY2VzIERpdmlzaW9uMRkwFwYDVQQDExBUaGF3dGUgU2VydMvYIENB
MSYwJAYJKoZIhvcNAQkBFhdzZXJ2ZXItY2VydHNAAGhhd3R1LmNvbTAeFw050TEy
MTIwNjM1NDBaFw0wMDEyMjUwNjM1NDBaMIGQMswCQQYDVQQGEwJVUzETMBEGA1UE
CBMKQ2FsaWZvcn5pYTEWMBQGA1UEBxMNU2FuIEZyYW5jaXNjbzEXMBUGA1UEChMO
THVjaWRpdHksIEluYy4xGjAYBgNVBAsTEWx1Y2lkZXN5Z2lmdHMUy29tMR8wHQYD
VQDEwZmdW5kLm1yY2lkZXN5Z2lmdHMUy29tMIGfMA0GCSqGSIb3DQEBAQUAA4GN
ADCBiQKBgQC+LdMchoCg14RcFfjTDQJFGiOJPMPU2w1SB/OCGYcgdIco91t8DdPd
Hxvvp56BJCkrn+gcPWLvSE1Nnw03n+/VsQMKNkbmPPKd8xsIOCx+oHEHtHEzfSEQ
aaibGjbcTaan1GFHEQzQD+bbeo2vwVbnJe3WXBKwnIGKyVGF6iUCcwIDAQABoyUw
IzATBgNVHSUEDDAKBggrBgEFBQcDATAMBgNVHRMBAf8EAjAAMA0GCSqGSIb3DQEBA
BAUAA4GBACdhXetCDrZ6fbVfbZJChOMUDKwO7XPIIhfHEM8V+eh6UYb9PbLGuYaO
pBwOExyLnQphstvsAdDf2f5twhKflwIH9uI52dpoZswh+1qbSdqM4REwwQqG9716
4Qho8BrMRr7r1K1X59WcuAPvGLG96lQgH2Rhetk0qSpB0d0Zpss0

-----END CERTIFICATE-----

14

```
<%doc>
Component to get the LOC based on the u_id

</%doc>
<%args>
$u_id
</%args>
<%init>
my( %data, $value );
my $giftcard = "10.20.1.53";
my $url      = "http://$giftcard/process/get_LOC?u_id=$u_id";
my $call_ret = mc_comp('/shared/http/lwp_call', url=>$url, data=>\%data );

if( $call_ret == 200 ) {
    $value = $data{'LOC'};
} else {
    $value = 0;
}

return $value;

</%init>
```

```
<%doc>
this component queries the database based on the passed arguments and determines
if card has been created for user.  If so, it returns the card number.  If not,
it returns a 0.
</%doc>
<%init>
my $card_no=0;
my $sql = 'select rcv_giftcard from orders where u_id=:1';
my $dbh = DBI->connect('DBI:Oracle:ligifts')
               or die "Couldn't connect to database: " . DBI->errstr;
my $sth = $dbh->prepare($sql)
               or die "Couldn't prepare statement: " . $dbh->errstr;

return $card_no;
</%init>
```

```
#!/usr/bin/perl

use strict;

use Apache::DBI;
use Apache::Constants qw(:common);

# Global DB connection
my ($dbh);

sub init_handler
{
    $dbh = DBI->connect('DBI:mysql:database=trail;host=localhost', '', '', {RaiseError => 1});
}

sub exit_handler
{
    $dbh->disconnect;
}

sub level1_handler
{
    my $r = shift;

    # Setup header and return blank gif
    $r->send_http_header('image/gif');
    open(IMG, '/local/lucidity/trail/htdocs/0/pixel.gif') or die "$$: Cannot open file: $!";
    $r->send_fd(*IMG);
    close(IMG);

    # Parse URI and setup notes for later logging
    my ($tmp, $action, $partner, $section, $uid, $amount) = split (/\/\//, $r->uri);
    my $query_string = $r->args; # or %query_string = $r->args;

    $r->notes('LUCD_ACTION' => $action);
    $r->notes('LUCD_PARTNER' => $partner);
    $r->notes('LUCD_SECTION' => $section);
    $r->notes('LUCD_UID' => $uid);
    $r->notes('LUCD_AMOUNT' => $amount);

    # Insert into DB if action accept
    if ($action eq 'accept') {
        my $sth = $dbh->prepare(qq/insert into user_clicks (action, partner, section, uid, amount, query_string) values ('$action', '$partner', '$section', '$uid', '$amount', '$query_string')/);
        $sth->execute;
        $sth->finish;
    }

    return OK;
}

1;
```

```
#!/usr/bin/perl

package HTML::Mason;

#
# Sample Mason handler.
#
use HTML::Mason;
use strict;

# Uncomment the next line if you plan to use the Mason previewer.
#use HTML::Mason::Preview;

# List of modules that you want to use from components (see Admin
# manual for details)
{
    package HTML::Mason::Commands;
    use CGI qw(:standard);
    use CGI::Cookie;
    use Date::Manip;
    use LWP::UserAgent;
    use HTTP::Cookies;
    use HTTP::Request;
    use HTTP::Response;
    use HTML::Entities;
    use Apache::Session::File;
}

# Create Mason objects
my $parser = new HTML::Mason::Parser;
my $interp = new HTML::Mason::Interp
(
    parser=>$parser,
    comp_root=>'/home/lucidity/front/fund/comps',
    data_dir=>'/home/lucidity/front/fund/mason'
);
my $ah = new HTML::Mason::ApacheHandler (interp=>$interp);
$ah->output_mode("stream");
$ah->error_mode("fatal");

# Activate the following if running httpd as root (the normal case).
# Resets ownership of all files created by Mason at startup. Change
# these to match your server's 'User' and 'Group'.
chown (scalar(getpwnam "nobody"), scalar(getgrnam "tech"),
    $interp->files_written);

sub handler
{
    my $r = shift;

    # If you plan to intermix images in the same directory as
    # components, activate the following to prevent Mason from
    # evaluating image files as components.
    #
    return -1 if $r->content_type && $r->content_type !~ m|^text/|io;

    # This block of code can be enabled to create a session-hash that every
    # component can access. This is useful for maintaining state across
```



```
# multiple requests. The Apache::Session module is required.
#
#my %session;
#my $cookie = $r->header_in('Cookie');
#$cookie =~ s/SESSION_ID=(\w*)/$1/;
#tie %session, 'Apache::Session::File', $cookie, {'Directory' => '/tmp/sessi
on'};
#$r->header_out("Set-Cookie" => "SESSION_ID=$session{_session_id};") if ( !$
cookie );

# This creates a global called %session that is accessible in all components
.
# Feel free to rename this as needed.
#
#local *HTML::Mason::Commands::session = \%session;

$ah->handle_request($r);

#untie %HTML::Mason::Commands::session;
}

1;
```

you made it to fund0!

ServerName fund.luciditygifts.com

ResourceConfig /dev/null
AccessConfig /dev/null

DefaultType text/html
#Options -Indexes

User nobody
Group tech

ServerRoot /home/lucidity/front/fund/servers
DocumentRoot /home/lucidity/front/fund/comps

Redirect /index.html http://www.lucidityinc.com/00/mainpage.html
Redirect /home http://www.lucidityinc.com/00/mainpage.html
Redirect /cgi-bin/vendor/gcfund https://fund.luciditygifts.com/giftcard/add

ErrorLog logs/error
LogFormat "%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-Agent}i\" \"%{cookie}n\" %T"
TransferLog logs/access.httpd

MaxClients 60
StartServers 50
MinSpareServers 40
MaxSpareServers 60
MaxRequestsPerChild 100

DirectoryIndex home index.html

PerlRequire /home/lucidity/front/fund/servers/conf/handler.pl
PerlTransHandler main::trans_handler

<Location />
 SetHandler perl-script
 PerlHandler HTML::Mason
 #PerlTypeHandler main::type_handler
 #PerlFixupHandler main::fixup_handler
</Location>

<Location /cgi-bin>
 SetHandler cgi-script
</Location>

<Location /ps>
 SetHandler perl-script
 PerlHandler Apache::Status
</Location>

```
ServerName fund.luciditygifts.com
ServerType standalone
```

```
Port 443
```

```
ResourceConfig /dev/null
AccessConfig /dev/null
```

```
DefaultType text/html
#Options -Indexes
```

```
Redirect /cgi-bin/vendor/gcfund https://fund.luciditygifts.com/giftcard/add
```

```
SSLEngine on
SSLVerifyClient 0
SSLVerifyDepth 10
SSLCertificateKeyFile /home/lucidity/front/fund/servers/conf/keys/server.key
SSLCertificateFile /home/lucidity/front/fund/servers/conf/keys/fund.crt
```

```
User nobody
Group tech
```

```
ServerRoot /home/lucidity/front/fund/servers
DocumentRoot /home/lucidity/front/fund/comps
```

```
ErrorLog logs/error
LogFormat "%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-Agent}i\" \"%{cookie}n\" %T"
TransferLog logs/access
```

```
MaxClients 60
StartServers 50
MinSpareServers 40
MaxSpareServers 60
MaxRequestsPerChild 100
```

```
DirectoryIndex home index.html
```

```
PerlRequire /home/lucidity/front/fund/servers/conf/handler.pl
#PerlTransHandler main::trans_handler
```

```
<Location />
    SetHandler perl-script
    PerlHandler HTML::Mason
    #PerlTypeHandler main::type_handler
    #PerlFixupHandler main::fixup_handler
</Location>
```

```
<Location /cgi-bin>
SetHandler cgi-script
</Location>
```

```
<Location /ps>
    SetHandler perl-script
    PerlHandler Apache::Status
</Location>
```

53

```
_map" SRC="charities/oxfam_det.gif" width="100" height="120" border="0"></td>
<td bgcolor="#cc99cc">&nbsp;</td>
<td bgcolor="#cc99cc" align="center" valign="middle"><IMG usemap="#so_ma
p" SRC="charities/so_det.gif" width="100" height="120" border="0"></td>
<td bgcolor="#cc99cc">&nbsp;</td>
</tr>
<tr>
<td colspan="13" bgcolor="#cc99cc" align="right"><a href="index2.html"><
img src="morecharities.gif" width="100" height="30" border="0"></a></td>
<td bgcolor="#cc99cc" align="center" valign="middle"></td>
</tr>
</table>
<table border="0" cellpadding="0" cellspacing="0" width="700">
<tr>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="260" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="260" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="20" bgcolor="#cc99cc">&nbsp;</td>
</tr>
<tr>
<td bgcolor="#cc99cc">&nbsp;</td>
<td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="
99ccff_cc99ccqlr15.gif" WIDTH="30" HEIGHT="30"></td>
<td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="aboutchariti
es.gif" WIDTH="260" HEIGHT="30"></td>
<td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="
99ccff_cc99ccq2r15.gif" WIDTH="30" HEIGHT="30"></td>
<td colspan="2" bgcolor="#cc99cc">&nbsp;</td>
<td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="
99ccff_cc99ccqlr15.gif" WIDTH="30" HEIGHT="30"></td>
<td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="howitworks.g
if" WIDTH="260" HEIGHT="30"></td>
<td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="
99ccff_cc99ccq2r15.gif" WIDTH="30" HEIGHT="30"></td>
<td colspan="2" bgcolor="#cc99cc">&nbsp;</td>
</tr>
<tr>
<td colspan="2" bgcolor="#cc99cc">&nbsp;</td>
<td colspan="2" bgcolor="#ffffff">&nbsp;</td>
<td colspan="3" height="200" bgcolor="#ffffff">&nbsp;</td>
<td colspan="2" bgcolor="#ffffff">&nbsp;</td>
<td colspan="2" bgcolor="#cc99cc">&nbsp;</td>
<td colspan="2" bgcolor="#ffffff">&nbsp;</td>
<td colspan="3" height="200" bgcolor="#ffffff">&nbsp;</td>
<td colspan="2" bgcolor="#ffffff">&nbsp;</td>
<td colspan="2" bgcolor="#cc99cc">&nbsp;</td>
</tr>
</table>
<tr>
```

```
<td bgcolor="#cc99cc">&nbsp;</td>
<td colspan="2" bgcolor="#cc99cc" align="center" valign="top"><IMG SRC="
fffff_cc99ccq3r15.gif" WIDTH="30" HEIGHT="30"></td>
<td bgcolor="#ffffff" align="center" valign="top">&nbsp;</td>
<td colspan="2" bgcolor="#cc99cc" align="center" valign="top"><IMG SRC="
fffff_cc99ccq4r15.gif" WIDTH="30" HEIGHT="30"></td>
<td colspan="2" bgcolor="#cc99cc">&nbsp;</td>
<td colspan="2" bgcolor="#cc99cc" align="center" valign="top"><IMG SRC="
fffff_cc99ccq3r15.gif" WIDTH="30" HEIGHT="30"></td>
<td bgcolor="#ffffff" align="center" valign="top">&nbsp;</td>
<td colspan="2" bgcolor="#cc99cc" align="center" valign="top"><IMG SRC="
fffff_cc99ccq4r15.gif" WIDTH="30" HEIGHT="30"></td>
<td bgcolor="#cc99cc">&nbsp;</td>
</tr>
<tr>
<td colspan="14" height="20" bgcolor="#cc99cc">&nbsp;</td>
</tr>
<tr>
<td colspan="2">&nbsp;</td>
<td colspan="2"></td>
</tr>
</table>
</center>
<map name="acs_map">
<area shape="rect" coords="25,0,75,20" href="charities/acs_det.html">
<area shape="rect" coords="25,100,75,120" href="charities/acs_sel.html">
</map>
<map name="egpaf_map">
<area shape="rect" coords="25,0,75,20" href="charities/egpaf_det.html">
<area shape="rect" coords="25,100,75,120" href="charities/egpaf_sel.html"
">
</map>
<map name="hsus_map">
<area shape="rect" coords="25,0,75,20" href="charities/hsus_det.html">
<area shape="rect" coords="25,100,75,120" href="charities/hsus_sel.html"
">
</map>
<map name="mawf_map">
<area shape="rect" coords="25,0,75,20" href="charities/mawf_det.html">
<area shape="rect" coords="25,100,75,120" href="charities/mawf_sel.html"
">
</map>
<map name="oxfam_map">
<area shape="rect" coords="25,0,75,20" href="charities/oxfam_det.html">
<area shape="rect" coords="25,100,75,120" href="charities/oxfam_sel.html"
">
</map>
<map name="so_map">
<area shape="rect" coords="25,0,75,20" href="charities/so_det.html">
<area shape="rect" coords="25,100,75,120" href="charities/so_sel.html">
</map>
</BODY>
</HTML>
```

```
<HTML><HEAD>
<TITLE>Blue Mountain Arts Charity Gift Center - 2</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" background="sky_bg.gif">
<center>
<table border="0" cellpadding="0" cellspacing="0" width="700">
<tr>
<td><IMG SRC="attcorner.gif" WIDTH="80" HEIGHT="73"></td>
<td bgcolor="#cc99cc"><IMG SRC="bmacharitytitle.gif" WIDTH="310" HEIGHT="
73"></td>
<td width="270" bgcolor="#cc99cc" align="left" valign="top">&nbsp;<p>bla
h text</p>
<td><IMG SRC="rtcorner.gif" WIDTH="40" HEIGHT="73"></td>
</tr>
</table>
<table border="0" cellpadding="0" cellspacing="0" width="700">
<tr>
<td width="15" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="100" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="14" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="100" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="14" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="100" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="7" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="7" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="100" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="14" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="100" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="14" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="100" height="5" bgcolor="#cc99cc">&nbsp;</td>
<td width="15" height="5" bgcolor="#cc99cc">&nbsp;</td>
</tr>
<tr>
<td bgcolor="#cc99cc">&nbsp;</td>
<td colspan="5" bgcolor="#cc99cc" align="left" valign="top"></td>
<td colspan="2" bgcolor="#cc99cc">&nbsp;</td>
<td colspan="5" bgcolor="#cc99cc" align="left" valign="bottom">Click "de
tails" to find out more about the charity.<p>
Click "select" to go directly to the donation page.</td>
<td bgcolor="#cc99cc">&nbsp;</td>
</tr>
<tr>
<td bgcolor="#cc99cc" colspan="14" height="14">&nbsp;</td>
</tr>
<tr>
<td bgcolor="#cc99cc">&nbsp;</td>
<td bgcolor="#cc99cc" align="center" valign="middle"><IMG usemap="#tnc_m
ap" SRC="charities/tnc_det.gif" WIDTH="100" height="120" border="0"></td>
<td bgcolor="#cc99cc">&nbsp;</td>
<td bgcolor="#cc99cc" align="center" valign="middle"><IMG usemap="#unice
f_map" SRC="charities/unicef_det.gif" WIDTH="100" height="120" border="0"></td>
<td bgcolor="#cc99cc">&nbsp;</td>
<td bgcolor="#cc99cc" align="center" valign="middle"><IMG usemap="#wwf_m
ap" SRC="charities/wwf_det.gif" width="100" height="120" border="0"></td>
<td colspan="2" bgcolor="#cc99cc">&nbsp;</td>
<td bgcolor="#cc99cc" align="center" valign="middle">&nbsp;</td>
<td bgcolor="#cc99cc">&nbsp;</td>
<td bgcolor="#cc99cc" align="center" valign="middle">&nbsp;</td>
<td bgcolor="#cc99cc">&nbsp;</td>
</tr>
```



```
<td bgcolor="#cc99cc" align="center" valign="middle">&nbsp;&nbsp;&nbsp;</td>
<td bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
</tr>
<tr>
  <td bgcolor="#cc99cc" align="center" valign="middle"></td>
  <td colspan="13" bgcolor="#cc99cc" valign="middle" align="left"><a href=
"index.html"></a
></td>
</tr>
</table>
<table border="0" cellpadding="0" cellspacing="0" width="700">
<tr>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="260" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="260" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td width="15" height="20" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
</tr>
<tr>
  <td bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="
p9ccff_cc99ccqlr15.gif" WIDTH="30" HEIGHT="30"></td>
  <td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="aboutchariti
es.gif" WIDTH="260" HEIGHT="30"></td>
  <td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="
99ccff_cc99ccq2r15.gif" WIDTH="30" HEIGHT="30"></td>
  <td colspan="2" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="
99ccff_cc99ccqlr15.gif" WIDTH="30" HEIGHT="30"></td>
  <td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="howitworks.g
if" WIDTH="260" HEIGHT="30"></td>
  <td colspan="2" bgcolor="#99ccff" align="center" valign="top"><IMG SRC="
99ccff_cc99ccq2r15.gif" WIDTH="30" HEIGHT="30"></td>
  <td colspan="2" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
</tr>
<tr>
  <td colspan="2" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td colspan="2" bgcolor="#ffffff">&nbsp;&nbsp;&nbsp;</td>
  <td colspan="3" height="200" bgcolor="#ffffff">&nbsp;&nbsp;&nbsp;</td>
  <td colspan="2" bgcolor="#ffffff">&nbsp;&nbsp;&nbsp;</td>
  <td colspan="2" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td colspan="2" bgcolor="#ffffff">&nbsp;&nbsp;&nbsp;</td>
  <td colspan="3" height="200" bgcolor="#ffffff">&nbsp;&nbsp;&nbsp;</td>
  <td colspan="2" bgcolor="#ffffff">&nbsp;&nbsp;&nbsp;</td>
  <td colspan="2" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
</tr>
<tr>
  <td colspan="2" bgcolor="#cc99cc">&nbsp;&nbsp;&nbsp;</td>
  <td colspan="2" bgcolor="#cc99cc" align="center" valign="top"><IMG SRC="
```

```
ffffff_cc99ccq3r15.gif" WIDTH="30" HEIGHT="30"></td>
    <td bgcolor="#ffffff" align="center" valign="top">&nbsp;</td>
    <td colspan="2" bgcolor="#cc99cc" align="center" valign="top"><IMG SRC="
ffffff_cc99ccq4r15.gif" WIDTH="30" HEIGHT="30"></td>
    <td colspan="2" bgcolor="#cc99cc">&nbsp;</td>
    <td colspan="2" bgcolor="#cc99cc" align="center" valign="top"><IMG SRC="
ffffff_cc99ccq3r15.gif" WIDTH="30" HEIGHT="30"></td>
    <td bgcolor="#ffffff" align="center" valign="top">&nbsp;</td>
    <td colspan="2" bgcolor="#cc99cc" align="center" valign="top"><IMG SRC="
ffffff_cc99ccq4r15.gif" WIDTH="30" HEIGHT="30"></td>
    <td bgcolor="#cc99cc">&nbsp;</td>
</tr>
<tr>
    <td colspan="14" height="20" bgcolor="#cc99cc">&nbsp;</td>
</tr>
<tr>
    <td colspan="2">&nbsp;</td>
    <td colspan="2"></td>
</tr>
</table>
```

```
</center>
<map name="tnc_map">
    <area shape="rect" coords="25,0,75,20" href="charities/tnc_det.html">
    <area shape="rect" coords="25,100,75,120" href="charities/tnc_sel.html">
</map>
<map name="unicef_map">
    <area shape="rect" coords="25,0,75,20" href="charities/unicef_det.html">
    <area shape="rect" coords="25,100,75,120" href="charities/unicef_sel.htm
">
</map>
<map name="wwf_map">
    <area shape="rect" coords="25,0,75,20" href="charities/wwf_det.html">
    <area shape="rect" coords="25,100,75,120" href="charities/wwf_sel.html">
</map>
</BODY>
</HTML>
```

```
<%$data%>
<%once>
use URI::Escape;
</%once>
<%init>
my @data;
foreach my $key (keys %ARGS) {
    if (ref $ARGS{$key} eq 'ARRAY') {
        foreach my $key2 (@{$ARGS{$key}}) {
            push @data, "$key=$key2";
        }
    } else {
        push @data, "$key=$ARGS{$key}";
    }
}
my $data = join "&", @data;
$data = uri_escape($data);
</%init>
```

```
<%init>
$response =~ s/card_number=\d+(\d{4})\&/card_number=...$1&/;
$url =~ s/card_number=\d+(\d{4})\&/card_number=...$1&/;
my $fh_lock = new IO::File ">>$logfile.lock" or return 0;
my $ctdown=5;
while (!HTML::Mason::Utils::get_lock($fh_lock)) {
    return 0 unless $ctdown--;    sleep 1;
}
my $LOG = new IO::File ">>$logfile";
my $data = << "EOF";
<LWP_CALL>
<START_DATE>
$start_date
</START_DATE>
<URL>
$url
</URL>
<RESPONSE>
$response
</RESPONSE>
<END_DATE>
$end_date
</END_DATE>
</LWP_CALL>
EOF

print $LOG $data;
return 1;
</%init>
<%cleanup>
$fh_lock->close;
</%cleanup>
<%ARGS>
logfile=>' /home/lucidity/front/www/data/lwp_log'
$start_date=>undef
$end_date=>undef
$url=>undef
$response=>undef
</%ARGS>
```

```
<%init>
my $logfile = '/home/lucidity/front/www_secure/data/' . $giftcat . '_error.log';
my $data_string;
$ARGS{card_number} =~ s/\d+(\d{4})/...$1/;
$url =~ s/card_number=\d+(\d{4})\&/card_number=...$1&/;
foreach my $key (keys %ARGS) {
    $data_string .= "$key => $ARGS{$key}\n";
}
my $fh_lock = new IO::File ">>$logfile.lock" or return 0;
my $ctdown=5;
while (!HTML::Mason::Utils::get_lock($fh_lock)) {
    return 0 unless $ctdown--;    sleep 1;
}
my $LOG = new IO::File ">>$logfile";
my $data_log = << "EOF";
<LWP_CALL>
<DATE>
$date
</DATE>
<URL>
$url
</URL>
<RETURN>
$return
</RETURN>
<DATA>
$data_string
</DATA>
</LWP_CALL>
EOF
print $LOG $data_log;
inc_comp('/shared/utls/send_mail', subject=>'$giftcat Error $date', text=>$data_log);
return 1;
</%init>
<%cleanup>
$fh_lock->close;
</%cleanup>
<%ARGS>
$giftcat=>'giftcard'
$return=>undef
$url=>undef
$date=>localtime
</%ARGS>
```

```
<%init>
$response =~ s/card_number\=\d+(\d{4})\&/card_number=...$1&/;
$url =~ s/card_number\=\d+(\d{4})\&/card_number=...$1&/;
my $fh_lock = new IO::File ">>$logfile.lock" or return 0;
my $ctdown=5;
while (!HTML::Mason::Utils::get_lock($fh_lock)) {
    return 0 unless $ctdown--;    sleep 1;
}
my $LOG = new IO::File ">>$logfile";
my $data = << "EOF";
<LWP_CALL>
<START_DATE>
$start_date
</START_DATE>
<URL>
$url
</URL>
<RESPONSE>
$response
</RESPONSE>
<END_DATE>
$end_date
</END_DATE>
</LWP_CALL>
EOF

print $LOG $data;
return 1;
</%init>
<%cleanup>
$fh_lock->close;
</%cleanup>
<%ARGS>
$logfile=>' /home/lucidity/front/inet-claim0/logs/lwp_log'
$start_date=>undef
$end_date=>undef
$url=>undef
$response=>undef
</%ARGS>
```

<%doc>

Will provide a connection to a specified url.

Will then store the contents in data depending
on how data was passed.

passing \%data will store the contents as a string.

passing \@data will store the contents as an array of name=value pairs.

passing \%data will store the contents as a hash of name/value pairs.

The component will return the reutn call of the lwp request.

</%doc>

<%init>

my \$Sdate = (localtime)[2].":".(localtime)[1].":".(localtime)[0];

my \$ua = new LWP::UserAgent;

\$ua->agent("Mozilla/4.7 [en] (Lucidity)");

\$ua->timeout(\$timeout);

my \$req = new HTTP::Request 'GET', \$url;

my \$res = \$ua->request(\$req);

my \$retry_count = 0;

Retry up to 3 times since it is NT that we are hitting...

while (\$res->code ne 200) {

\$retry_count ++;

if (\$retry_count gt 3) { return \$res->code; }

\$res = \$ua->request(\$req);

}

my \$Edate = (localtime)[2].":".(localtime)[1].":".(localtime)[0];

mc_comp ('/shared/utis/log_call', start_date=>\$Sdate, end_date=>\$Edate, url=>\$u

rl, response=>\$res->content);

if (ref \$data eq 'ARRAY') {

(@{\$data}) = split "&", \$res->content;

} elsif (ref \$data eq 'HASH') {

my (@tmp) = split "&", \$res->content;

foreach my \$key (@tmp) {

my (\$name, \$value) = split "=", \$key, 2;

\$data->{\$name} = \$value;

}

} else {

\$\$data = \$res->content;

}

return \$res->code;

</%init>

<%ARGS>

\$url

\$data

\$timeout=>5

\$retry=>3

</%ARGS>

<%DOC>

Will provide a connection to a specified url.
Will then store the contents in data depending
on how data was passed.

passing \%data will store the contents as a string.

passing \%@data will store the contents as an array of name=value pairs.

passing \%data will store the contents as a hash of name/value pairs.

The component will return the reuthn call of the lwp request.

</%DOC>

<%ARGS>

\$url

\$data

\$content=>''

\$timeout=>30

\$retry=>3

\$method=>'GET'

\$details=>0

</%ARGS>

<%ONCE>

use LWP::UserAgent;

</%ONCE>

<%INIT>

#----- Get start time for logging purposes -----#

my \$start_date = (localtime)[2].":".(localtime)[1].":".(localtime)[0];

#----- Create user agent -----#

my \$ua = new LWP::UserAgent;

\$ua->agent("Mozilla/4.7 [en] (Lucidity)");

\$ua->timeout(\$timeout);

#----- Request the URL -----#

my \$req = new HTTP::Request \$method, \$url;

if(\$method eq 'POST'){

my(\$a, \$b) = split(/\?/, \$url);

\$url = \$a if(\$a);

\$content = \$b unless(\$content);

\$req->content_type('application/x-www-form-urlencoded');

\$req->content(\$content);

my \$res = \$ua->request(\$req);

#----- Retry number of times requested by the user -----#

my \$retry_count = 0;

while((\$res->code != 200) && (\$retry_count <= \$retry)) {

\$res = \$ua->request(\$req);

\$retry_count ++;

}

#----- Get end time and log call -----#


```
my $end_date = (localtime)[2].":".(localtime)[1].":".(localtime)[0];
mc_comp ( '/shared/utils/log_lwp_call',
          start_date=>$start_date, end_date=>$end_date,
          url=>$url, response=>$res->content,
          logfile=>'/home/lucidity/front/inet-claim0/logs/chocolate/lwp_calls.l
og' );

#----- Return data -----#
if( ref $data eq 'ARRAY' ) {
    (@{$data}) = split "&", $res->content;
} elsif( ref $data eq 'HASH' ) {
    my (@tmp) = split "&", $res->content;
    foreach my $key (@tmp) {
        my( $name, $value ) = split "=", $key, 2;
        $data->{$name} = $value;
    }
} else {
    $$data = $res->content;
}

if( $details ) {
    my $result;
    $result->{'code'} = $res->code;
    $result->{'message'} = $res->message;
    $result->{'headers'} = $res->headers;
    $result->{'content'} = $res->content;
    return( $result );
} else {
    return( $res->code );
}

/%INIT>
```

<%doc>

This will return a call from an lwp request with the correct name/value pair string.

syntax:

<& /shared/http/lwp_return, status=>0, msg=>"database down" &>

</%doc>

<& /shared/utils/info_string, %ARGS &>

td

```
<%ONCE>
use URI::Escape;
</%ONCE>
<%INIT>
my @data;

foreach my $key (keys %ARGS) {
    if( ref $ARGS{$key} eq 'ARRAY' ) {
        foreach my $value (@{$ARGS{$key}}) {
            push @data, "$key=$value";
        }
    } else {
        push @data, "$key=$ARGS{$key}";
    }
}

my $data = join "&", @data;

$data = uri_escape($data);

return( $data );

</%INIT>
```

[illegible]

This file controls what MIME types are sent to the client for the
given file extensions. Sending the correct MIME type to the client
is important so they know how to handle the content of the file.
Extra types can either be added here or by using an AddType directive
in your config files. For more information about MIME types
please read RFC 2045, 2046, 2047, 2048, and 2077.

# MIME type	Extension
application/activemessage	
application/andrew-inset	
application/applefile	
application/atomicmail	
application/dca-rft	
application/dec-dx	
application/luciditykey	key
application/mac-binhex40	hqx
application/mac-compactpro	cpt
application/macwriteii	
application/msword	doc
application/news-message-id	
application/news-transmission	
application/octet-stream	bin dms lha lzh exe class
application/oda	oda
application/pdf	pdf
application/postscript	ai eps ps
application/powerpoint	ppt
application/remote-printing	
application/rtf	rtf
application/slate	
application/smil	smi smil sml
application/wita	
application/wordperfect5.1	
application/x-bcpio	bcpio
application/x-cdlink	vcd
application/x-compress	
application/x-cpio	cpio
application/x-csh	csh
application/x-director	dcr dir dxr
application/x-dvi	dvi
application/x-gtar	gtar
application/x-gzip	
application/x-hdf	hdf
application/x-javascript	js
application/x-koan	skp skd skt skm
application/x-latex	latex
application/x-mif	mif
application/x-netcdf	nc cdf
application/x-sh	sh
application/x-shar	shar
application/x-stuffit	sit
application/x-sv4cpio	sv4cpio
application/x-sv4crc	sv4crc
application/x-tar	tar
application/x-tcl	tcl
application/x-tex	tex
application/x-texinfo	texinfo texi
application/x-troff	t tr roff
application/x-troff-man	man
application/x-troff-me	me
application/x-troff-ms	ms

application/x-ustar	ustar
application/x-wais-source	src
application/zip	zip
audio/basic	au snd
audio/midi	mid midi kar
audio/mpeg	mpga mp2 mp3
audio/x-aiff	aif aiff aifc
audio/x-pn-realaudio	ram
audio/x-pn-realaudio-plugin	rpm
audio/x-realaudio	ra
audio/x-wav	wav
chemical/x-pdb	pdb xyz
image/gif	gif
image/ief	ief
image/jpeg	jpeg jpg jpe
image/png	png
image/tiff	tiff tif
image/x-cmu-raster	ras
image/x-portable-anymap	pnm
image/x-portable-bitmap	pbm
image/x-portable-graymap	pgm
image/x-portable-pixmap	ppm
image/x-rgb	rgb
image/x-xbitmap	xbm
image/x-xpixmap	xpm
image/x-xwindowdump	xwd
message/external-body	
message/news	
message/partial	
message/rfc822	
model/iges	igs iges
model/vrml	wrl vrml
model/mesh	msh mesh silo
multipart/alternative	
multipart/appledouble	
multipart/digest	
multipart/mixed	
multipart/parallel	
text/css	css
text/html	html htm
text/plain	txt
text/richtext	rtx
text/tab-separated-values	tsv
text/x-setext	etx
text/x-sgml	sgml sgm
text/xml	xml dtd
video/mpeg	mpeg mpg mpe
video/quicktime	qt mov
video/x-msvideo	avi
video/x-sgi-movie	movie
x-conference/x-cooltalk	ice

This is a comment. I love comments.

```
<%PERL_ARGS>
$first_month=>'1'
$show_many=>'12'
$selected=>' '
$name=>'month'
$style=>'num'
</%PERL_ARGS>
<%PERL_INIT>
$show_many = ($show_many > 12 ? 12 : $show_many);
```

```
my %names = (
    1 => 'January',
    2 => 'Febuary',
    3 => 'March',
    4 => 'April',
    5 => 'May',
    6 => 'June',
    7 => 'July',
    8 => 'August',
    9 => 'September',
    10 => 'October',
    11 => 'November',
    12 => 'December',
);
```

```
</%PERL_INIT>
```

```
<SELECT NAME="<%$name%>">
  <OPTION VALUE="">Month
  <%
    foreach my $month (1 .. $show_many) {
      my $month_num = sprintf "%2.2d", $month;
      my $month_name = $names{ $month };
      if( $selected == $month_num ) {
        <OPTION VALUE="<%$month_num%>" SELECTED><%( $style eq 'name' ? $month_name
: $month_num )%>
      } else {
        <OPTION VALUE="<%$month_num%>"><%( $style eq 'name' ? $month_name : $month
_num )%>
      }
    }
  </SELECT>
```

```
<%init>
my (%data)      = mc_comp('mall_lookup', u_id=>$u_id);
my (%trail_data) = mc_comp('trail_lookup', u_id=>$u_id);
my $amount      = $data{balance}-$trail_data{amount};
$amount         = $amount ? $amount : 0;
return $amount;
</%init>
<%args>
$u_id
</%args>
```



```
<%args>
$url
</%args>
<%init>
$r->send_cgi_header("Location: $url\n\n");
</%init>
```

```
<%perl_doc>
this component is used to send a return payment request to CyberCash CashRegiste
r
</%perl_doc>
<%args>
$order_id
$amount
</%args>
<%once>
use CyberC::CCMckLib3_2;
use CyberC::CCMckDirectLib3_2;
use CyberC::CCMckErrno3_2;

# Initialize the Global Configuration array in CCMckLib3_2
# JW - change setting of $ConfigFile variable based on where it will eventually
reside and confirm giftcat names
my $ConfigFile;
if ($ARGS{giftcat} eq 'charities'){
    $ConfigFile = '/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/con
f/merchant_conf';
}
elsif ($ARGS{giftcat} eq 'giftcert'){
    $ConfigFile = '/home/www/servers/lucidity/app/lucidityinc-83/mck-cgi/con
f/shine_conf';
}
my $status = CyberC::CCMckLib3_2::InitConfig($ConfigFile);
</%once>
<%init>
my (%result) = CyberC::CCMckDirectLib3_2::SendCC2_1Server('return', %ARGS);
return (\%return);
</%init>
```

MIIC5zCCA1CgAwIBAgIDAKATMA0GCSqGSIb3DQEBAUAMIHEMQswCQYDVQQGEwJaQTEvMBMGAlUECBMMV2ZsdG9ybiBDYXBlMRIeAYDVQQHEw1DYXBlIFRvdzI2XHTABGtGNVBA0TFRROXYd0ZSdb25zdW0aW5nIGNjMgSwJGxYDVQQLXExDZXJ0aWZpY2F0aW9uIFN1cnZpY2VzIERpdmlzaW9uMRkwfWYDVQQJExBAUGF3dGUGU2V2dmVpYIENBMSYwJAYJKoZIhvcNAQkBFhdzZXJ2ZXItY2VydHNAAdGhhd3RlLmNvbTAeFw050TExMTgxNzUyNTBAFw0mMDByMDExNzUyNTBAMIGEMQswCQYDVQQGEwJVUzERMA8GA1UeChBMIRG5YXdhcmUeXzARBGtGNVBA0TClpbpG1pbmd0b245FjAUBGtGNVBA0TDXUxY2lkacXR15LCBjbmMxeEjAQBGNVBA0TCDV1Ym1hc3RlcjEhBM8GA1UEAxMyc2VjZjXJlLmx1Y2lkaxR5Z2ZlmdHMuY292MTGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC+LdMchoCg14RcfjTDQJFGi0JPMKPu2lSB/OCGYcgdIco91t8DdP6Hxv1vp56BJCkrmn+gCpWLvSE1N1xw03n+/VsQMMKbmPPKd8xsIOcx+oHEtHezfSeQaaiBgjbcTAA5N1GFHEQZQD+bbec2vwVbnJe3WXBKwnIGKYVGF6iUCwC1DAQAB0YUz1ATBGNVH5UEDDAKBggrBgEFBQCcDATAMBgNVHRMBAf8EAjAAMA0GCSqGSIb3DQEBAUAA4GBADtkQToXFwHdcdrtD/m58aZyFg2uBzo99dY2VbVr+wk0C9yHWuelK7jnhg9Uu+ResVEi115ShyDFwHd2vpDkZ5dFxiYR20UacD+E61Dgx4ziwZ8HybWodQKIX7Tg61Zf54Phb0GMFFN6HytWElp2bnxLlIbf0sUi8jNtSrZkToRdev

-----END CERTIFICATE-----

75

```
<%init>
my($fullDate);
$dest_address =~ s/\s+\/\, /g;
chomp($fullDate = `date`);
# Do we want to append/prepend to text at all
# Create the mail command
my($mail_cmd) = "To: $dest_address\nFrom: $reply_to\nSubject: $subject\n";
$mail_cmd .= "Reply-to:$reply_to\n";
if ($bounce_to) {
    $mail_cmd .= "Bounce-to:$bounce_to\n";
}
$mail_cmd .= "Error:$fullDate\n$text\n";
open (SENDMAIL, "| /usr/lib/sendmail -t");
print SENDMAIL $mail_cmd;
close(SENDMAIL);
```

```
</%init>
<%ARGS>
$dest_address=>'fatalerror@ligifts.com'
$text=>'An Error occurred'
$subject=>'Error ' . localtime
$reply_to=>'tech@ligifts.com'
$bounce_to=>'tech@ligifts.com'
</%ARGS>
```

```
%foreach my $key (keys %ARGS) {  
    <% $key %> : <% $ARGS{$key} %><BR>  
}%
```

```
<%PERL_ARGS>
$selected=>' '
$name=>' '
</%PERL_ARGS>
<%PERL_INIT>
my @states = mc_comp( '/shared/data/states' );

</%PERL_INIT>

<SELECT NAME="<%%$name%>">
  <OPTION VALUE="">Select state
  % foreach my $state (@states) {
  %   if( $selected eq $state->{'code'} ) {
  %     <OPTION VALUE="<%%$state->{'code'}%>" SELECTED><%%$state->{'name'}%>
  %   } else {
  %     <OPTION VALUE="<%%$state->{'code'}%>"><%%$state->{'name'}%>
  %   }
  % }
</SELECT>
```

```
#!/usr/bin/perl

use HTML::Mason;
use strict;

my $outbuf;
my $parser = new HTML::Mason::Parser;
my $interp = new HTML::Mason::Interp( parser=>$parser,
                                       comp_root=>'/home/www/comps/shared/utils',
                                       data_dir=>'/home/www/mason/shared/utils',
                                       out_method=>\$outbuf );

my $retval = $interp->exec( '/unique_id.new' );

open( F, "mason.out" );
print F $outbuf;
close(F);
print "return value of component was: $retval\n";
```

```
Btime=<% (localtime)[2] %>:<% (localtime)[1] %>&Etime=<% (localtime)[2] %>:<% (localtime)[1] %>
```

[illegible]

80


```
<%ARGS>
$u_id
</%ARGS>
<%INIT>
my $trailp = "10.20.1.54";
my $url = "http://$trailp/push/get_amount?u_id=$u_id";
my %data;
my $return = mc_comp('/shared/http/lwp_call', url=>$url, data=>\%data);
return %data;
</%INIT>
```

```
<%doc>
Process to generate unique UUIDs. This algorithm will continue to work properly u
ntil the year 2587 :)
Based on GMT time and the process ID number ($$)
</%doc>
<%init>
my( $ID,$hex );
my( $sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst ) = gmtime(time());
my $rand = int(rand(10));

# create ID based on date information and process number
# $yday = day of the year (0-365); $yday+1 to get the more familiar 1-366
# $year is years since 1900 (3 digits)
# $msec = seconds since midnight (5 digits, since it's at most 86400)

my $msec = ($hour*3600)+($min*60)+$sec;
my $num = sprintf( "%03d%03d%05d", $year, $yday+1, $msec );

# Convert to hex to compress number from 11 to 9 characters while keeping it uni
que
# (hex function produces overflow so this is done explicitly)

my $test = sprintf( "%9x", $num );

my( $i );

my $temp = $num;
while( $temp >= 16 ) {
    $temp = $temp/16;
    $i++;
}

while( $i>=0 ) {
    my $exp = 16**$i;
    my $value = int($num/$exp);
    if( $value == 10 ) {
        $value = 'A';
    } elsif( $value == 11 ) {
        $value = 'B';
    } elsif( $value == 12 ) {
        $value = 'C';
    } elsif( $value == 13 ) {
        $value = 'D';
    } elsif( $value == 14 ) {
        $value = 'E';
    } elsif( $value == 15 ) {
        $value = 'F';
    }
    $hex = $hex . $value;
    $num = $num - ($exp*$value);
    $i--;
}

while( length($hex) < 9 ) {
    $hex = '0' . $hex;
}

# Add PID and a random number

$ID = sprintf( "%09d%05d%01d", $hex,$$, $rand );
```

```
return $ID;  
</%init>
```

```
<%doc>
borrowed code from CyberCash to generate unique id's
based on GMT time and the process ID number ($$)
</%doc>
<%init>
my($ID);
my($sec,$min,$hour,$mday,$mon,$year,$yday,$isdst) = gmtime(time());

# create ID based on date, time, and process number
# $mon is the month index where Jan=0 and Dec=11, so we use
# $mon+1 to get the more familiar Jan=1 and Dec=12
# $year is years since 1900: we want only two digits

$ID = sprintf("%02d%02d%02d%02d%05d", $year % 100, $mon+1,$mday,$hour,$min,$
$);

return $ID;
</%init>
```

```
<%doc>
borrowed code from CyberCash to generate unique id's
based on GMT time and the process ID number ($$)
</%doc>
<%init>
my($ID);
my($sec,$min,$hour,$mday,$mon,$year,$yday,$isdst) = gmtime(time());

# create ID based on date, time, and process number
# $mon is the month index where Jan=0 and Dec=11, so we use
# $mon+1 to get the more familiar Jan=1 and Dec=12
# $year is years since 1900: we want only two digits

$ID = sprintf("%02d%02d%02d%02d%05d", $year % 100, $mon+1,$mday,$hour,$min,$
$);

return $ID;
</%init>
```

```
<%ARGS>
$u_id
</%ARGS>
<%INIT>
my (%data, %trail_data);
my $giftcard = "10.20.1.53";
my $url = "http://$giftcard/process/return_int_vars?u_id=$u_id";
my $return = mc_comp('/shared/http/lwp_call', url=>$url, data=>\%data);
$data{return_int_vars} = $return;

my $trailp = "10.20.1.54";
$url = "http://$trailp/push/get_amount?u_id=$data{u_id}";
$return = mc_comp('/shared/http/lwp_call', url=>$url, data=>\%trail_data);
$data{trail_data} = $return;

my $balance      = $data{balance} - $trail_data{amount};
$data{db_balance} = $data{balance};
$data{tr_balance} = $trail_data{amount};
$balance          = 0 if $balance < 0;
$data{balance}    = $balance;

return %data;
</%INIT>
```

```
%foreach my $key (keys %ARGS) {
    <% $key %>=<% $ARGS{$key} %><BR>
%}
```

87

```
<%init>
my (@data, %data, $key);
my $data = mc_file($logfile);
my $first = 1;
my $array_count = 0;
foreach my $line (split "\n", $data) {
    next if $first && ($first = ($line =~ /\<$delim\>/) ? 0 : 1);
    if ($line =~ m#<$delim>#) {
        $key = undef;
        next;
    } elsif ($line =~ m#</$delim>#) {
        $array_count++;
        $key = undef;
        next;
    } elsif ($line =~ m#^<(\w+)>$#) {
        $key = $1;
    } elsif ($line =~ m#^</(\w+)>$#) {
        $key = undef;
    } else {
        $data[$array_count]->{$key} .= $line."\\n" if $key;
    }
}
</%init>
<HTML><BODY>
%foreach my $href (@data) {
    if ($start_date) {
        next unless $href->{START_DATE} =~ /^$start_date$/;
        foreach my $id (keys %{$href}) {
            <% $id %>: <% $href->{$id} %><BR>
        }
    } else {
        Time Of Call:<A HREF="/admin/view_log?logfile=<%$logfile%>&start_date=<
$href->{START_DATE} %>"><% $href->{START_DATE} %></A><BR>
    }
}
</BODY> </HTML>
<%ARGS>
$logfile=>' /home/lucidity/front/www_secure/data/lwp_log'
$delim=>'LWP_CALL'
$start_date=>undef
</%ARGS>
```



```
<%PERL_ARGS>
$first_year=>'
$show_many=>'20'
$selected=>'
$name=>'year'
</%PERL_ARGS>
<%PERL_INIT>
$first_year = 1999;      # Replace with call to get year...

</%PERL_INIT>

<SELECT NAME="<$$name%>">
  <OPTION VALUE=" ">Year
  % for( my $year=$first_year ; $year<($first_year+$show_many) ; $year++ ) {
  %   if( $selected == $year ) {
  %     <OPTION VALUE="<$$year%>" SELECTED><$$year%>
  %   } else {
  %     <OPTION VALUE="<$$year%>"><$$year%>
  %   }
  % }
</SELECT>
```

gift

Copy 1 of 3

[illegible]

```
<%ARGS>
$LOC
</%ARGS>
<%ONCE>
use DBI;
use Lucidity::DBITools;
</%ONCE>
<%INIT>
my( $order );
my $dbh = Lucidity::DBITools::dbi_connect('production');

#----- Get information about gift certificate order -----#
my $sqlstmt = "select
                TO_CHAR( o.purchase_date, 'Month DD, YYYY' ) as purchase_date,
                o.to_name, o.to_email, o.from_email, o.amount, o.card_first_name,
                o.card_last_name
            from
                bma_orders o
            where
                o.loc = '$LOC';

Lucidity::DBITools::dbi_exec_for_row_hash( $dbh, $sqlstmt, \%order );

#----- Pick out the first name if we can -----#
my( $to_first_name, $to_last_name );
if( $order{ to_name } =~ /\s+/ ) {
    ( $to_first_name, $to_last_name ) = split( / /, $order{ to_name } );
} else {
    $to_first_name = $order{ to_name };
}

my $order_amount = sprintf ( "%4.2f", $order{ amount } );

</%INIT>

Dear <%$order{ card_first_name }%>,

Thank you for sending a Bluemountain Gold Gift Certificate.

We received your order on <%$order{ purchase_date }%>.

An e-mail notification has been sent to <%$order{ to_name }%> at <%$order{ to_email }%>.

You ordered: Bluemountain Gold Gift Certificate

Item Price: $<%$order_amount%>
Quantity: 1
Subtotal: $<%$order_amount%>

ORDER TOTAL: $<%$order_amount%>

If any of the above information is incorrect, please e-mail support@bluemountaingifts.com to modify your order so that we may correct your order. Be sure to mention that you are a Bluemountain sender.
```

If you want to view general information on Bluemountain gift certificate cards, visit: <http://www.luciditygifts.com/giftcard/infopage>.

Bluemountain - helping the world give and communicate

[illegible]

Hi,

I did some seriouse re-arranging in here. Most old comps are now in the SAV dir

PLease make charge the comp that charges credit cards (calls cybercash)

"create" is a place holder for where the card creation stuff will go.

Dennis